Desarrollo de Arquitecturas para Procesamiento de Control de la Mirada en Sistemas de Visión Activa con Resolución Espacial Variable

Tesis Doctoral

Martín González García

Escuela Técnica Superior de Ingeniería de Telecomunicación

2016

AUTOR: Martín González García
http://orcid.org/0000-0002-9216-0372

EDITA: Publicaciones y Divulgación Científica.
Universidad de Málaga

Departamento de Tecnología Electrónica
E.T.S.I. de Telecomunicación
Universidad de Málaga

Tesis Doctoral

Desarrollo de Arquitecturas para Procesamiento de Control de la Mirada en Sistemas de Visión Activa con Resolución Espacial Variable

Autor:
Martín González García
Ingeniero de Telecomunicación

Director:
Pelegrín Camacho Lozano
Doctor Ingeniero de Telecomunicación

Don Pelegrín Camacho Lozano, Doctor Ingeniero de Telecomunicación

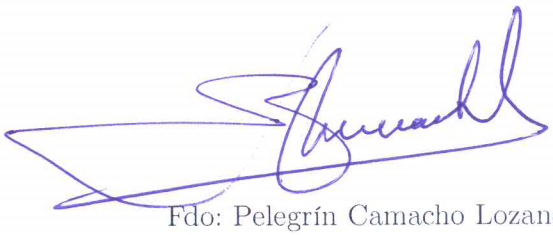
Informa:

Que D. Martín González García, Ingeniero de Telecomunicación, ha realizado en el Departamento de Tecnología Electrónica de la Universidad de Málaga, bajo su dirección el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

DESARROLLO DE ARQUITECTURAS PARA PROCESAMIENTO DE CONTROL DE LA MIRADA EN SISTEMAS DE VISIÓN ACTIVA CON RESOLUCIÓN ESPACIAL VARIABLE

Revisado el presente trabajo, estima que puede ser presentado al Tribunal que ha de juzgarlo, por lo que AUTORIZA y AVALA la presentación de esta Tesis en la Universidad de Málaga.

Málaga, a 11 de Noviembre de 2015



Fdo: Pelegrín Camacho Lozano
Dr. Ingeniero de Telecomuniación

Resumen

En esta Tesis se aborda la implementación de un sistema completo de visión activa, en el que se capturan y generan imágenes de resolución espacial variable. Todo el sistema se integra en un sólo dispositivo del tipo AP SoC (*All Programmable System on Chip*), lo que nos permite llevar a cabo el codiseño hardware-software del mismo, implementando en la parte lógica los bloques de preprocesado intensivo, y en la parte software los algoritmos de procesado de control más complejo. El objetivo es que, trabajando con un campo visual del orden de Megapíxeles, se pueda procesar una tasa moderada de imágenes por segundo. Las imágenes multiresolución se generan a partir de sensores de resolución uniforme con una latencia nula, lo que permite tener preparada la imagen de resolución variable en el mismo instante en que se ha terminado de capturar la imagen original. Como innovación con respecto a las primeras contribuciones relacionadas con esta Tesis, se procesan imágenes con toda la información de color. Esto implica la necesidad de diseñar conversores entre espacios de color distintos, para adecuar la información al tipo de procesado que se va a realizar con ella. Estos bloques se integran sin alterar la latencia de entrega de los sucesivos fotogramas. El procesamiento de estas imágenes multirresolución genera un mapa de saliencia que permite mover la fóvea hacía la región considerada como más relevante en la escena.

El contenido de la imagen se estructura en una jerarquía de niveles de abstracción. A diferencia de otras arquitecturas de este tipo, como son la pirámide regular y el polígono foveal, en las que se trabaja con imágenes de resolución uniforme en los distintos niveles de la jerarquía, la pirámide irregular foveal que se propone en esta Tesis combina las ideas de trabajar con un imagen realmente multirresolución, que incluya el campo de visión completo que abarcan sensor y óptica, con el procesamiento jerárquico propio de las pirámides irregulares. Para ello en esta Tesis se propone la implementación de un algoritmo de diezmado irregular que, tomando como base la imagen multirresolución, dará como resultado una estructura piramidal donde los distintos niveles no son imágenes sino grafos orientados a la resolución del problema de segmentación y estimación de saliencia.

Todo el sistema se integra en torno a la arquitectura de bus AXI, que permite conectar entre si todos los cores desarrollados en la parte lógica, así como el acceso a la memoria compartida con los algoritmos implementados en la parte software. Esto es posible gracias a los bloques de acceso directo a memoria AXI-VDMA, en una propuesta de configuración que permite tanto la integración perfectamente coordinada de la transferencia de la imagen multirresolución generada a la zona de trabajo del algoritmo de segmentación como su recuperación para la posterior visualización del resultado del proceso, y todo ello con una tasa de trabajo que mejora los resultados de plataformas similares.

Abstract

This Thesis proposes the development of an active vision system, which is able to capture and generate space-variant images. All the system is endowed within a unique AP SoC (All Programmable System-on-chip), following the guidelines of the hardware-software codesign: those blocks in charge of the intensive pre-processing are synthesised on the logic part (hardware), meanwhile the algorithms that require a more complex control are programmed on the software part. The aim is that, working with a visual field of the order of megapixels, the whole system can process a moderate rate of images per second. Multiresolution images are generated from a CMOS sensor of large resolution with zero latency. This allows the system to provide the space-variant image when the capture of the original image ends. One of the novelties with respect to the first contributions related to this PhD Thesis is that images are processed considering all the colour information. This implies the design of converters able to translate between different colour spaces. Thus, we adapt the information to the type of processing that will be performed with her. These blocks are integrated within the architecture without altering the latency delivery of successive frames. The processing of these multiresolution images focuses on generating a saliency map. This map allows to continuously moving the fovea to the region considered the most relevant one in the scene.

For obtaining the saliency map, the image content is structured into a hierarchy of levels of abstraction. Unlike other similar proposals, such as regular pyramids or foveal polygons, in which all levels of the hierarchy are encoded as images of uniform resolution, the irregular foveal pyramid proposed in this Thesis successfully combines the idea of working with a multiresolution image, which covers the visual field provided by sensor and optics, with the hierarchical processing coming from irregular pyramids. The result is a novel decimation scheme that deals with a hierarchy whose levels are encodes using simple graphs.

The system is built around the AXI bus. It allows to connect all the cores development on the logic part of the AP SoC, and also that the algorithms running on its software part can access to the shared memory. This is possible thanks to the blocks of direct memory access AXI-VDMA, in an architectural proposal that permits both the perfectly coordinated integration of the transfer of the multiresolution images to the working area of the segmentation algorithm and its recovery for the rear display of the obtained results. And all this with a frame rate that improves the results obtained on similar platforms.

Tabla de contenido

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El término visión variante en el espacio aparece a finales de la década de los 80s para referirse a aquellas arquitecturas visuales que se basan en la variación, sobre el campo visual capturado, de la resolución de muestreo de datos. Son los también denominados sistemas de visión foveales, haciendo así referencia a la fóvea, la zona del ojo humano que captura la escena a máxima resolución. Asociados al desarrollo de soluciones que permitan percibir la escena usando cámaras y procesadores limitados, los sistemas de visión foveales se integrarán muchas veces con sistemas activos de control de la mirada y pares binoculares o cabezas robóticas (ver Figura 1.1). Es la época en la que se rompe con la denominada visión constructivista, asentada en los postulados de David Marr[1], y en la que se sugiere que la percepción visual debe resolver el problema de describir escenas, para plantear que la visión debe servir, junto con otros sentidos, para interactuar con el entorno y sobrevivir en él - evitar obstáculos, reconocer y coger objetos…-. Es por ello la época de las cabezas binoculares o estéreo, los sistemas de control de la mirada, o los sensores de muestreo log-polar. También de los algoritmos jerárquicos de procesado visual, surgidos en muchos casos del mecenazgo de Azriel Rosenfeld y su *Computer Vision Laboratory* en la Universidad de Rutgers. Se plantea que el problema de la visión no debe tratar de resolverse

1 '*The goal of an image-understanding system is to transform two-dimensional data into a description of the three-dimensional spatiotemporal world … must infer 3D surfaces, volumes, boundaries shadows, occlusion, depth…*' (J.K. Tsotsos, 1997, p. 389)

buscando que un modelado matemático permita obtener buenos resultados prácticos, sino que el proceso puede ser el contrario. Esto es, que la observación y el trabajo en cómo capturar la información permita la construcción de teorías exitosas. Es una propuesta precursora del *embodiment*, tan de boga en la robótica actual: el modelado matemático que, en la década de los 70s y principios de los 80s no había resuelto casos prácticos en visión, no podía desarrollarse ajeno ni al propio agente que percibe, y sus limitaciones hardware, ni tampoco a la tarea final a resolver (Aloimonos y Rosenfeld, 1991).

Figura 1-1: La cabeza robótica binocular Popeye, diseñada en el marco de la visión activa en el *Institute for Systems & Robotics*, Universidad de Coimbra, Portugal

Este marco, guiado en cualquier caso por una idea que, como escribe Peter Meer en 2001, era considerada en los años 60 como una tarea trivial: el diseño de un sistema genérico y autónomo de percepción visual, pierde fuerza conforme se entra en el nuevo siglo. El desarrollo de sensores foveales o sistemas de procesamiento jerárquico no parecen realmente necesarios ante la capacidad de procesamiento de los nuevos ordenadores personales, y se impone con rotundidad la necesidad de trabajar en soluciones muy acotadas por la aplicación, ahora en escenarios en los que las soluciones que surgen de la Estadística y el Cálculo probabilístico se imponen (como harán posteriormente en robótica (Thrun et al, 2005)). La posibilidad de nuevos sensores visuales de cada vez mayor tamaño y el advenimiento de las nuevas plataformas de sistemas en chip totalmente programables (AP SoC) abren nuevas posibilidades para el desarrollo de sistemas de captura variante en el espacio. Montadas sobre plataformas binoculares, estas nuevas arquitecturas pueden convertirse en una alternativa a las populares cámaras RGBD.

1.1. Objetivos de la Tesis

El objetivo de esta Tesis es el diseño e implementación de un sistema de percepción visual sobre una plataforma AP SoC. Dicho sistema incluirá:

- un módulo de adquisición de imagen foveal, que emulará el funcionamiento de un sensor de geometría cartesiano-exponencial de fóvea desplazable y tamaño variable;
- un módulo de segmentación de imagen, que generará una jerarquía de niveles de abstracción creciente sobre los datos extraídos del sensor; y
- un módulo de estimación de *saliencia* basado en descriptores obtenidos de las regiones en que se divide la imagen, que permitirá guiar la fóvea para englobar la región que se considere como más relevante.

Los dos primeros módulos configuran la implementación de la que denominaremos pirámide irregular foveal, una propuesta de cambio en la organización horizontal de los datos de entrada a una pirámide irregular que, aprovechando la flexibilidad del grafo, supone que la base sea una imagen de resolución no uniforme. Esto rompe con la definición de pirámide como una jerarquía de niveles de resolución uniforme, pues los algoritmos de diezmado siempre aportarán más detalle a las zonas de cada nivel que se vayan generando sobre la fóvea.

Todo el diseño se lleva a cabo para procesar el flujo de información a la mayor velocidad posible pero embebiendo la solución en un marco software-hardware plausible. Esto implicará abordar numerosos problemas y la toma de decisiones de diseño en los que se balanceará calidad, recursos hardware y tiempo de proceso.

1.2. Motivación

Desde hace ya algún tiempo, la alta capacidad de integración de la tecnología CMOS ha permitido desarrollar e integrar una amplia gama de aplicaciones o sistemas en el mismo chip. Los denominados SoC (*System on Chip*) no sólo suponen un avance en lo que se refiere al hardware sino que además, al incorporar elementos de procesamiento para el desarrollo software, suponen un escenario distinto para el diseño y desarrollo de arquitecturas complejas, que requieren de las ventajas ofrecidas tanto por el hardware sintetizado como de la programación software. Por otra parte, no sólo los nuevos

microprocesadores empotrados siguen sacando provecho del desarrollo de la Ley de Moore, y su asociado aumento en la densidad de integración y potencia computacional, sino que ésta también afecta directamente al diseño de los sensores visuales, cada vez de mayor resolución. En este nicho surge el concepto de cámara inteligente, como un dispositivo que integra sensor y procesador empotrado para extender la capacidad típica de la cámara, la de capturar fotos o secuencias de vídeo, para que ahora sea capaz de procesar estos datos y así, incluso tomar decisiones en tiempo real. De esta forma, se puede entender que la cámara es capaz de trabajar a un medio o alto nivel, sirviendo la información relativa por ejemplo a la presencia de una persona o el seguimiento de un objeto en movimiento.

El grupo de Ingeniería de Sistemas Integrados de la Universidad de Málaga tomará contacto con la temática de la visión activa en aquellos primeros años de los 90s, cuando la visita de César Bandera conducirá a nuestro grupo al restringido marco de las retinotopologías cartesiano-exponencial, la pirámide regular de Peter Burt o el polígono foveal. Fruto del trabajo en esa dirección serán las propuestas de desplazamiento de la fóvea en el campo de visión, la adaptación de su tamaño al objeto de interés o las poco explotadas retinotopologías multifoveales, propuestas en los sucesivos trabajos de Fabián Arrebola y Pelegrín Camacho. El procesamiento jerárquico será la base de la implementación a medida de la pirámide regular propuesta en la Tesis Doctoral de Francisco J. Coslado, pero también de la propuesta irregular presentada en la Tesis Doctoral de Rebeca Marfil. Aunque se habían esbozado propuestas en trabajos previos en el propio grupo, esta última Tesis propondrá también un mecanismo de atención que ha ido evolucionando hasta su versión más reciente, en el que se cierra el lazo percepción-acción incluyendo planificación automática, como se documenta en la Tesis Doctoral de Antonio J. Palomino. Los conceptos teóricos de sensado variante en el espacio, procesado jerárquico y mecanismo de atención forman parte, por tanto, del bagaje de nuestro grupo desde hace ya más de dos décadas. Pero además, en paralelo a este trabajo teórico, el grupo ha llevado a cabo un interesante recorrido en lo que se refiere al diseño de soluciones empotradas de visión artificial, basadas fundamentalmente en el empleo de dispositivos *Field Programmable Gate Arrays* (FPGA).

En 2014 el grupo propone una primera arquitectura en la que se combinan sensado variante en el espacio, pirámides irregulares y mecanismos de atención, en un trabajo que se publica en la revista *Frontiers* (Marfil et al., 2014). Aprovechando parcialmente el trabajo previo a estas fechas en todos estos campos, pero con el ánimo de implementar una arquitectura completa y también novedosa sobre un AP SoC surge esta Tesis Doctoral, cuyo resultado

será una cámara inteligente, capaz de mover la fóvea de forma autónoma a aquella región que sea considerada por la cámara como más relevante.

1.3. Propuesta del sistema de visión foveal empotrado

El esquema del sistema de visión foveal empotrado que se propone en la presente Tesis Doctoral se muestra en la Figura 1.2. El sistema incluye un sensor color de 5 Mpíxeles y un AP SoC, en el que se integrarán el hardware de captura de una imagen de fóvea desplazable y de tamaño optimizado y los algoritmos encargados de las segmentación de la imagen (usando un esquema de pirámide irregular) y de la evaluación de la saliencia para, en el marco general de un proceso de atención, poder determinar cuál es el objeto más relevante en la escena. En el esquema se ha incluido también la memoria externa, elemento básico para almacenar la enorme cantidad de datos con los que se trabaja. En la Figura 1.3 se muestran los elementos concretos con los que finalmente se llevarán a cabo las pruebas documentadas en esta Tesis: la placa Zedboard y su Zynq-7020 de Xilinx y el sensor OV5642 de Omnivision.

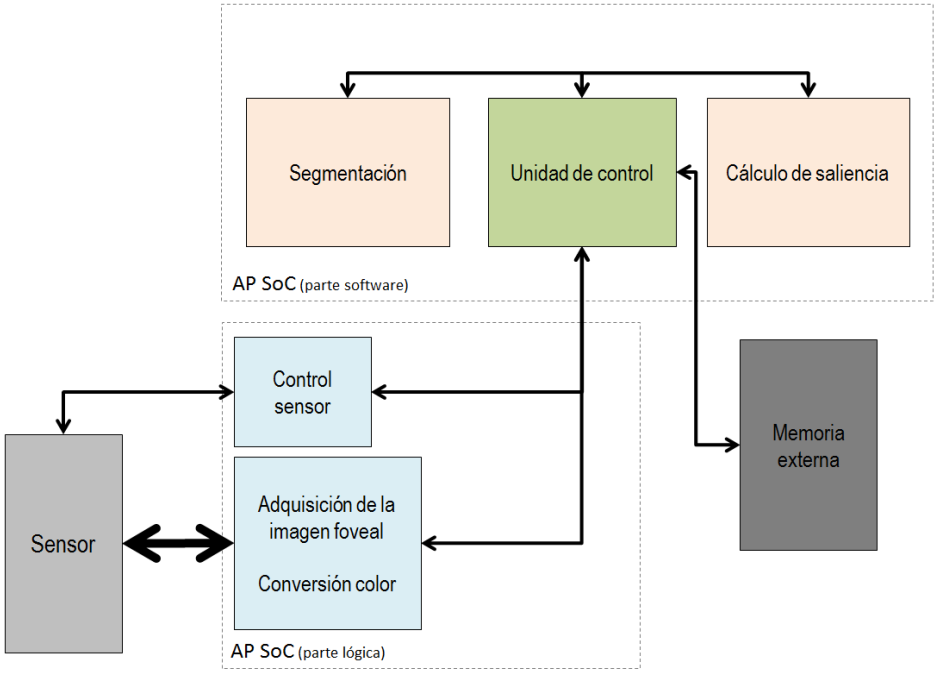


Figura 1-2: Esquema general del sistema propuesto

Figura 1-3: La placa Zedboard, empleada como entorno de desarrollo básico en la presente Tesis Doctoral, y el sensor OV5642

La arquitectura así definida está, sin embargo, incompleta. La información contenida en la fóvea no será procesada por ningún módulo de más alto nivel, por lo que el reconocimiento del contenido en la misma no influirá en el mecanismo de atención. Tampoco se incluye en éste el sistema de inhibición de retorno, necesario para evitar que la fóvea enmarque continuamente una misma región de la escena. Finalmente, la selección de los pesos de los distintos mapas de evidencias empleados para el cálculo de la saliencia o relevancia de cada región no serán modulados por la tarea en curso. Sin embargo, la arquitectura dispondrá de los interfaces necesarios para su conexión con todos estos módulos.

1.4. Contribuciones

El trabajo que finalmente se recoge en esta Tesis Doctoral comienza con la implementación hardware de las primeras propuestas de sensado variante en el espacio que se proponen en el marco del grupo de investigación. A continuación se recogen las contribuciones del autor de esta Tesis, incluyendo una breve descripción de las mismas.

- Francisco J. Coslado, Pelegrín Camacho, Martín González, Fabián Arrebola, Francisco Sandoval Hernández: VLSI Implementation of a Foveal Polygon Segmentation Algorithm. ICIAP: 185-190 (1999)

Este artículo presenta una arquitectura VLSI que implementa un algoritmo de segmentación en el marco del polígono foveal cartesiano-exponencial. El diseño permitía trabajar a 20-30 fotogramas/segundo.

- Pelegrín Camacho, Francisco J. Coslado, Martín González, F. Sandoval: Adaptive multiresolution imager based on FPGAs. EUSIPCO (2000)

  Esta contribución analiza la implementación de las estrategias de sensado variante en el espacio en FPGAs, adaptadas para trabajar tanto con cámaras CCD como con sensores CMOS.

- Pelegrín Camacho, Francisco J. Coslado, Martín González, F. Sandoval: Multifoveal imager for stereo applications. Int. J. Imaging Systems and Technology 12(4): 149-165 (2002)

  En este artículo se analiza el uso de las propuestas foveales en el marco de las aplicaciones estéreo. La propuesta analiza el empleo de varias fóveas sobre un mismo campo de visión, así como la integración de algoritmos de sustracción de fondo para detección de movimiento o mecanismos de atención.

- Martín González, Pelegrín Camacho, Francisco J. Coslado, F. Sandoval: A Real Time Multiresolution Image Generator Implemented on a FPGA. XVII Conference on Design of Circuits and Integrated Systems: 113-118 (2002)

  En este artículo se presenta el diseño e implementación en una FPGA de bajo coste de una arquitectura para obtener imágenes multirresolución a partir de un sensor CMOS en escala de gris y hasta 1 megapixel de resolución. El diseño integrado propone también el interfaz con los bufferes de memoria.

- Francisco J. Coslado, Martín González, Pelegrín Camacho, Francisco Sandoval Hernández: Hardware platform for regions extraction in foveal images. VCIP: 1730-1740 (2003)

  El artículo analiza el procesamiento jerárquico para la segmentación de imagen y propone un diseño capaz de procesar 33 fotogramas/segundo.

- Martín González, José R. Salinas, Francisco J. Coslado, Pelegrín Camacho, Francisco Sandoval Hernández: Hierarchical test pattern composition to testing a foveal imager ASIC. VLSI Circuits and Systems: 497-505 (2003)

  Esta contribución propone una metodología jerárquica para generar conjuntos de patrones de test, para la verificación de un ASIC, tan reducidos como sea posible sin pérdida significativa en la cobertura de fallos. La técnica propuesta no emplea modelos abstractos para representar el problema, requiriendo sólo información estructural.

- Francisco J. Coslado, Pelegrín Camacho, Martín González, Francisco Sandoval Hernández: Hardware architecture for hierarchical segmentation in foveal images. Int. J. Imaging Systems and Technology 14(4): 153-166 (2004)

  Descripción del algoritmo de segmentación de imagen diseñado e implementado en el marco de la Tesis Doctoral de Francisco J. Coslado. La plataforma propuesta trabaja a velocidades comprendidas entre los 25 y los 85 fotogramas/segundo.

Dichas publicaciones culminan en 2004, con la defensa de la Tesis Doctoral de Francisco J. Coslado. Posteriormente, el grupo en visión foveal se desarticula, cuando tanto Fabián Arrebola como Pelegrín Camacho se apartan de esta línea de investigación. El trabajo del autor de esta Tesis Doctoral continúa, pero ahora en una línea más centrada en la docencia. En cierta forma, la Tesis Doctoral supone un reencuentro con los trabajos publicados hace ya más de diez años. En ese tiempo todo ha cambiado, tanto a nivel teórico, advenimiento de las pirámides irregulares y la representación de la imagen como un grafo, como a nivel de implementación final, con el surgimiento de las nuevas plataformas AP SoC o los sensores CMOS de alta resolución y calidad.

1.5. Estructura de la Tesis

Además del Capítulo de Introducción, la memoria de la presente Tesis Doctoral se estructura en seis Capítulos más. En el Capítulo 2 se describe con mayor detalle el escenario que rodea este trabajo. Se justifica la captura de imagen con resolución variante en el espacio en sistemas visuales como el humano y se presenta el problema de replicar un módulo de movimiento de dicho sistema como el que rodea los globos oculares humanos. Las bases biológicas permiten esbozar las soluciones que, para la captura variante en el espacio, se han propuesto desde el campo de la visión artificial. Se analizan los modelos

log-polar y geométrico-cartesiano, las propuestas que, en este último, se han sugerido para desplazar la fóvea o adaptarla a la región de interés. También los intentos por implementar los modelos en un hardware específico de captura. Además, el Capítulo revisa las aproximaciones jerárquicas al problema del procesado de la información visual, describiendo muy superficialmente las soluciones regulares e irregulares. Se aborda el esquema del polígono foveal y la nueva propuesta, planteada en esta Tesis Doctoral, de la pirámide irregular foveal.

El Capítulo 3 analiza los dos grandes bloques de procesamiento de bajo o medio nivel que se empotran en la arquitectura: el segmentador y el mecanismo de atención. El primero de ellos se plantea como una modificación de un método ya aplicado para la segmentación color de imágenes en el marco de las pirámides irregulares: el diezmado dirigido por los datos (D3P). Su implementación en el marco de esta Tesis obliga a reconsiderar determinadas características del proceso, para conseguir finalmente su implementación en el AP SoC. El mecanismo de atención, por su parte, trabaja con una segmentación de la imagen de entrada y un número limitado de regiones o proto-objetos. Su implementación software es muy similar a las ya desarrolladas en el seno del grupo de trabajo. Los algoritmos se estudian y presentan, en este Capítulo, desde el punto de vista de su funcionamiento y, aunque se modifican para su posterior implementación en el AP SoC, este aspecto se abordará fundamentalmente en el Capítulo 4.

El Capítulo 4 se centra en la implementación en el AP SoC del sistema completo. Comienza por tanto describiendo la adquisición del grafo que sirve de base a la pirámide irregular foveal, lo que implica el diseño e implementación del módulo de generación de la fóvea y anillos de resolución de la geometría cartesiano-exponencial de fóvea desplazable y tamaño generalizado. La resolución de esta fase está embebida en el hardware disponible para resolverlo, por lo que implica un análisis de las características de la parte lógica de la AP SoC. Se analiza el tratamiento del color, que también es una novedad frente a las propuestas anteriores, que siempre trabajan en niveles de gris. El segmentador se ubica entre las partes lógicas y software del AP SoC, mientras que el mecanismo de atención se implementa exclusivamente en la parte software.

El Capítulo 5 presenta los resultados del sistema completo y supone la validación de la arquitectura (la verificación parcial de los distintos módulos se aborda en los Capítulos 3 y 4). Se presentan las características técnicas de la implementación final, los tiempos de procesamiento y el análisis cuantitativo de los resultados obtenidos. El proceso completo se estructura como un sistema

de exploración de escenas, que produce finalmente una segmentación de la imagen de entrada. Esto permite su comparación con otros algoritmos de segmentación usando bases conocidas y populares de imágenes naturales.

Finalmente, el Capítulo 6 incluye las conclusiones, en lo que podríamos considerar como el cierre de la implementación proporcionada en esta Tesis Doctoral, y una discusión abierta de la propuesta, tanto en su análisis de qué es necesario hacer en concreto para mejorarla, como abordando el marco general en que se encuadra y las posibilidades que, en dicho marco, se abren con ella, tanto en líneas de trabajo como en posibilidades nuevas para el grupo de trabajo.

Capítulo 2

Hacia una retina artificial: la pirámide irregular de fóvea desplazable

Cuando estamos despiertos, con los ojos abiertos, las personas tenemos la impresión de que podemos tener una visión total, y en detalle, del mundo que se mueve frente a nosotros. Pero esto es sólo una impresión. La evolución de los sistemas de visión biológicos ha llegado a establecer un equilibrio entre lo que podemos percibir y lo que somos capaces de procesar en todo momento de forma rápida. Como bien explica Collin Ware (*Visual Thinking for Design*, 2008), esta ilusión de poder ver todo lo que tenemos frente a nosotros proviene del hecho de que podemos extraer cualquier detalle del mundo a través de la visión, en el momento que queramos, mediante un movimiento de los ojos que es, literalmente, más veloz que el pensamiento o la propia conciencia. Como describe muy gráficamente el psicólogo Kevin O'Regan, el mundo '*es su propia memoria*' (O'Regan, 1992).

Sin embargo, frente a esa necesidad de imagen de detalle, en la que los ojos se mueven por la escena para capturar la información deseada en función de la tarea[2], el sistema también precisa mantener una constancia de qué está

2 Como describe Alfred J. Yarbus en sus trabajos publicados en inglés en 1967 '*Records of eye movements show that the observer's attention is usually held only by certain elements of the picture.... It is easy to determine from these records which elements attract the observer's eye (and, consequently, his thought), in what order, and how often*' (Yarbus, 1967, p. 190)

ocurriendo con el resto de la escena, quizás a menor resolución, pero sí con el suficiente como para poder actuar en caso de necesidad. Frente a la visión central, que respondería a las habilidades de percepción en detalle hasta ahora comentadas, la denominada visión periférica es la habilidad de localizar, reconocer y responder a la información en las distintas áreas del campo visual alrededor del objeto sobre el cual se fija la atención. La retina periférica es especialmente sensible a los desplazamientos, siendo su función más característica la detección del movimiento, aunque también se ha constatado su utilización durante acciones de alcanzar y atrapar. También parece jugar un importante papel en la coordinación visuo-motora, la postura y locomoción en el espacio.

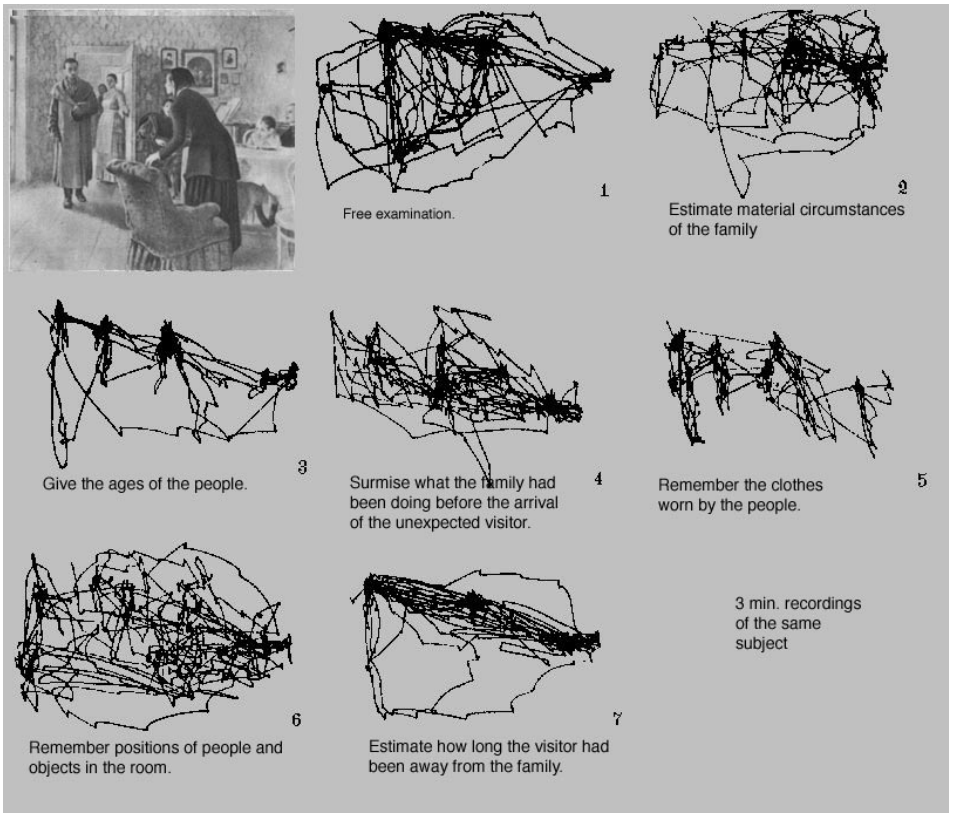


Figura 2-1: Dependencia con la tarea del movimiento seguido por los ojos frente a una determinada imagen (Yarbus, 1967)

En este Capítulo se describen las bases teóricas del diseño de una arquitectura de visión artificial que imita esta captura a distintas resoluciones de la imagen. En ocasiones, esta imitación tratará de acercarse al propio funcionamiento de nuestro sistema de visión. En otras, sin embargo, se alejará de éste por

razones de cómputo o eficiencia. Para ello, el Capítulo se organiza con una primera y muy breve introducción al sistema y proceso de visión humano (Sección 2.1). Después, en la Sección 2.2, se presentarán las aproximaciones artificiales a dicho sistema, no sólo en lo que respecta al sensor de captura, sino también a las primeras etapas del sistema de procesado de la información. En nuestro caso, el objetivo es implementar una arquitectura visual que permita el procesamiento jerárquico de una retinotopología foveal y en la que incluiremos un mecanismo artificial de atención. La novedosa arquitectura jerárquica propuesta en esta Tesis combina la captura de una imagen variante en el espacio con el concepto de jerarquía propio de las pirámides irregulares. La pirámide irregular foveal se introduce brevemente en la Sección 2.3, siendo su diseño e implementación descrito en el resto de Capítulos de esta Tesis.

2.1. El proceso de la visión humana

Los ojos son como pequeñas cámaras digitales, que contienen lentes que enfocan la imagen sobre el globo ocular. Aunque muchos encuentren un problema el hecho de que, como muestra la Figura 2.2, la imagen capturada esté invertida en la parte posterior del ojo, hay que tener en cuenta que, en este símil, el cerebro es como un ordenador, aunque bastante diferente eso sí del digital basado en silicio, y para él es tan fácil trabajar con una imagen invertida como con una que no lo esté.

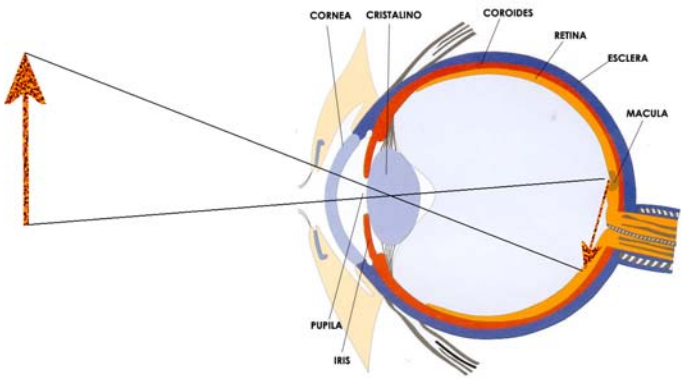


Figura 2-2: Formación de la imagen en la retina

Siguiendo con el símil, al igual que la cámara digital color tiene una matriz de elementos fotosensibles, que le permiten capturar la iluminación recibida a tres longitudes de onda distintas, el ojo humano tiene también una matriz de

receptores, los denominados conos, para capturar también tres colores distintos[3]. La analogía entre ambos sistemas va incluso más allá, pues al igual que las cámaras digitales comprimen la información antes de transmitirla o almacenarla, también la retina cuenta con varias capas de células que extraen lo que es más interesante de los datos capturados. Como resultado de este proceso, la información extraída de los más de 100 millones de receptores del ojo, puede ser transmitida al cerebro mediante el millón de fibras que, aproximadamente, forman el nervio óptico. En cualquier caso, ésta es sólo la primera etapa del sistema neuronal más complejo de todos nuestros sistemas sensoriales. Desde la retina, el eje óptico mueve la información al cerebro medio y el tálamo. Y desde esa zona el flujo de datos se mueve a la corteza visual primaria (Kandel et al., 2000).

Existen, sin embargo, profundas diferencias entre ambos sistemas. Mientras que las cámaras digitales emplean mayoritariamente sensores de resolución uniforme, en los que los fotorreceptores se reparten en una matriz, la retina presenta una distribución no uniforme, en la que los fotorreceptores se concentran en una región central, la denominada fóvea. Además, en el caso del ojo no es totalmente correcto hablar de fotorreceptores, pues éstos son sólo una parte del sistema. En su conjunto, se puede entender que los hasta seis tipos distintos de células que forman las tres capas de la retina (bipolares, amacrinas, ganglionares… (Bouldac y Levine, 1997)) se organizan como pequeños procesadores de imagen (Purves et al, 2001).

Como muestra la Figura 2.3, en nuestra retina únicamente una pequeña fracción de la escena, la recogida por la fóvea y los anillos adyacentes, es procesada a alta resolución espacial, mientras que la resolución con la que se procesa el resto de la escena, regiones intermedia y periférica, es bastante más baja. Esto es, el nivel de detalle con el que se capturan las imágenes no es constante a lo largo y ancho de campo de visión. De esta forma, la mitad de los recursos que el cerebro emplea en procesar la información visual sólo tratan con menos del 5% del campo visual percibido. Esto obliga a que el acto de percibir una escena implique el movimiento continuo de la fóvea, para que ésta incluya cada una de las zonas de interés en la misma. Dado que la estructura multirresolución de la retina es fija, este movimiento de la fóvea sólo puede ser llevado a cabo si se mueven los ojos. Esto es sumamente importante, y por ello el conjunto de músculos que mueven cada ojo (Figura 2.4) permiten que éste se desplace a unos 900 grados por segundo y entonces pararlo, todo en menos de una décima de segundo. Durante este instante de tiempo, conocido como movimiento sacádico, se suprime la visión.

3 Dejamos de lado los bastones, que principalmente trabajan sólo a niveles bajos de iluminación.

Así, aunque no seamos para nada conscientes de ello, percibimos el mundo a través de un conjunto de movimientos rápidos y fijaciones cortas. Controlar estos movimientos es evidentemente un aspecto clave del proceso de la visión en los seres humanos, que también es extensible a otros animales que, como los conejos, no presentan visión foveal.

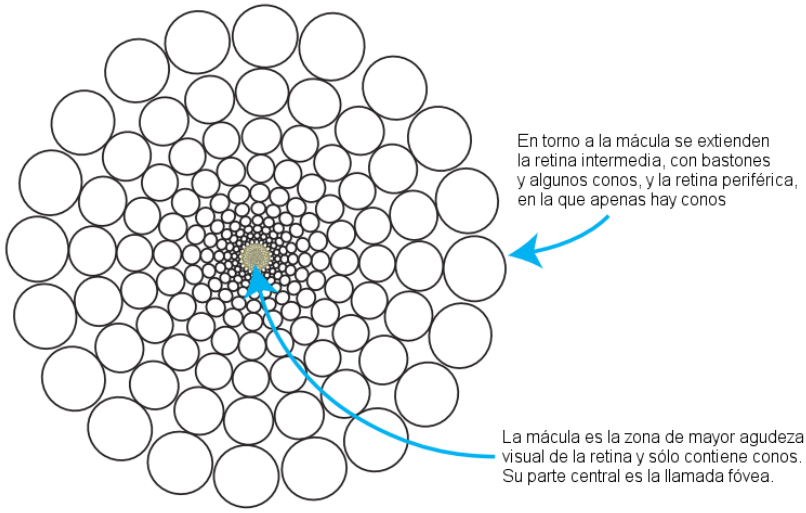


Figura 2-3: La distribución de los conos varía enormemente en la retina, dando lugar a *brain pixels* de tamaño variable, que se distribuyen siguiendo un modelo polar (adaptación del original de C. Ware, 2008, p. 5).

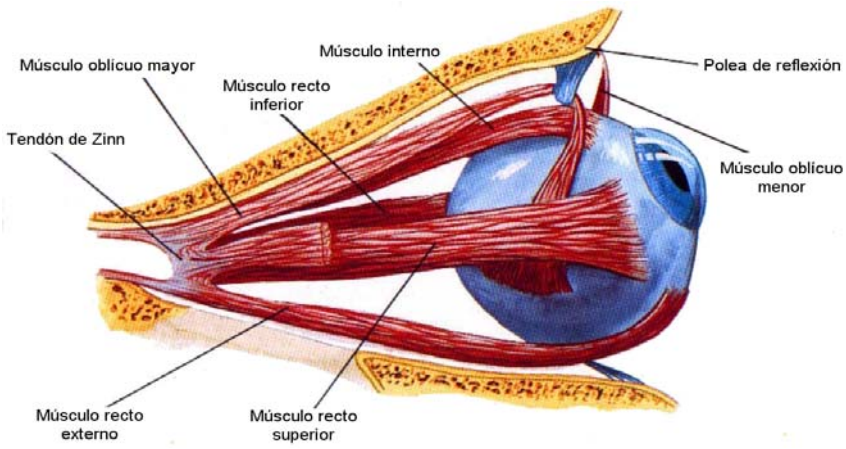


Figura 2-4: Distribución de músculos encargados del movimiento del ojo

Resulta complejo diseñar un control realimentado tradicional para movimientos tan rápidos, pues esto requeriría usar ganancias altas y estimar con mucha precisión las derivadas que el modelo engloba. En suma, esto da lugar a controladores que fácilmente se vuelven inestables (Lesmana y Pai, 2011). Los estudios llevados a cabo tanto en humanos como en animales evidencian que el controlador empleado podría modelarse como un *pulse-step* no lineal (Collins, 1975).

2.2. Sistemas artificiales de visión con muestreo variable en el espacio

El objetivo de los sistema artificiales de visión multirresolución es imitar el comportamiento no uniforme que presenta el sistema visual de algunos seres vivos. Este acercamiento se remonta a la década de los 70s, cuando ciertos investigadores en visión artificial deciden imitar la naturaleza foveal del sistema visual de los primates como una alternativa a los sensores de resolución uniforme convencionales. Y vendrá motivado, fundamentalmente, por los avances en la investigación de las propiedades espaciales del mapeo retino-cortical en los mamíferos, especialmente en los primates, que se llevan a cabo desde la década de los 40s. En 1960 se demuestra que el desplazamiento de un estímulo de luz incidente en la retina de estos sistemas provoca, como respuesta, un desplazamiento en la información que llegaba a la corteza visual, y que este desplazamiento era inversamente proporcional a la distancia a la fóvea (Daniel y Whitteridge, 1961). Este efecto (*cortical magnification*) se traduce en un factor μ_c, medido en milímetros de corteza por grado de ángulo visual, que permitirá caracterizar la transformación o mapeo de los datos visuales desde las coordenadas en la retina a las de la corteza visual. Empíricamente, se encontró que dicho factor podía aproximarse por (Schwartz, 1977).

$$\mu_c(\rho)=\frac{C_1}{1+C_2\rho} \tag{2.1}$$

donde ρ es la excentricidad en la retina, medida en grados, y C_1 y C_2 dos constantes que deben ser determinadas experimentalmente. Integrando esta ecuación, es posible establecer una relación entre la distancia cortical r y la excentricidad

$$r(\rho)=\int_0^{\rho}\frac{C_1}{1+C_2\rho}d\rho=\frac{C_1}{C_2}\log(1+C_2\rho) \tag{2.2}$$

Lo que permite observar el comportamiento general del mapeo retino-cortical, en el que el tamaño y distancia entre los campos receptivos asociados a cada célula ganglionar aumentaban linealmente con la excentricidad, esto es, con la distancia a la fóvea (Lindeberg y Florack, 1994). Además, también se completaron estudios que demostraban como estímulos lineales procedentes de la fóvea se organizaban aproximadamente a lo largo de líneas en la corteza visual, mientras que estímulos circulares centrados en la fóvea producían respuestas lineales en la corteza con orientaciones aproximadamente ortogonales (Tootell et al, 1982). Esto evidenciaba que el mapeo de la información que, transmitida desde la retina, llega a la corteza visual podía ser convenientemente expresado como una transformación conforme, esto es, como una función que presenta la propiedad de preservar los ángulos relativos[4]. En concreto, dicho mapeo podía modelarse, aproximadamente, siguiendo una ley logarítmico-polar (Schwartz, 1977). Esta será la base teórica del mapeo o transformación log-polar, sobre el cual volveremos en la Sección 2.2.1 del presente Capítulo.

El paradigma de la *visión activa* surgirá en este marco a finales de la década de los 80s y principios de la de los 90s. No debe confundirse sin embargo con la visión foveal o multirresolución (*space-variant*), cuyas bases hemos introducido en los párrafos anteriores. La visión activa incide en la necesidad de mover el sensor, para así dirigir el campo de visión del mismo al objeto de interés. Surge en un momento en el que un fotograma de 512x512 píxeles de tamaño ya era considerado elevado, por lo que controlar un campo de visión grande a una resolución adecuada sólo podía hacerse en un tiempo razonable si el sensor, de una resolución reducida, se movía, recorriendo así la escena. En este contexto aparecen los mecanismos de atención y de control de la mirada, sobre los que volveremos en el Capítulo 3 de esta Tesis, pues ahora no se trata sólo de reconocer un elemento de la escena, sino también de saber cuál es el objeto o zona de la imagen que debe ser visitada. Se postula que esta capacidad de mover el sensor permitirá aportar soluciones más eficientes a problemas que, desde la perspectiva pasiva o reconstructivista, eran no lineales y estaban mal definidos (Aloimonos et al., 1988). Al esquema se unirá la posibilidad de manipulación que ofrecían los brazos y manos robóticas (Bajcsy, 1993; Swain y Stricker, 1993), o también la de crear campos visuales virtuales de alta resolución (Schwartz et al., 1995). En este contexto, al incorporar al sistema de visión activa un sensor foveal, normalmente log-polar, surgirá el concepto de visión activa variante en el espacio (*space-variant active*

4 Matemáticamente, una función ω=log(z), donde ω y z son variables complejas, es conforme en el punto z si es holomorfa (o antiholomorfa) en z y su derivada en z es distinta de cero. Una función se dice holomorfa en un punto z si su derivada existe en z y en todos los puntos vecinos a z

vision), en el que sí se unen visión activa y captura multirresolución (Schwartz et al., 1995). Como se ha descrito en el Capítulo 1 de esta Tesis, este es el marco en el que se encuadra la presente Tesis.

A continuación se analizan en más detalle los dos modelos de transformación retino-cortical más empleados en visión artificial: el fundamentado en los trabajos introducidos en este apartado (el log-polar o log(z)) y el cartesiano. También se analizarán brevemente otras propuestas. Finalmente se analizarán los trabajos destinados a la fabricación específica de sensores no uniformes.

2.2.1. Transformación log-polar

El mapeo log-polar es una transformación geométrica de la imagen que intenta emular la reorganización de la información visual que, como se ha comentado brevemente anteriormente, se produce en los primates al mover ésta desde la retina a la corteza visual. Como se ha introducido en el apartado anterior, conceptualmente, el modelo retino-cortical log(z) considera la retina como un plano complejo, en el que el centro de la fóvea corresponde al origen, y la corteza visual como otro plano complejo. Las posiciones en la retina se representan por la variable compleja z, y las posiciones en la corteza por la variable compleja Ω. La correspondencia entre los dos planos viene dictado por la función Ω = log(z). Dicha función presenta una singularidad en el origen (z = 0), lo que complicará la emulación o fabricación del sensor (Figura 2.5).

Para eliminar el problema de la singularidad y fabricar un sensor físico, Giulio Sandini y su equipo introducirán dos mapeos distintos, uno para la fóvea -uniforme- y otro para la periferia -log-polar- (Sandini y Tagliasco, 1980; Braccini et al., 1981; Braccini et al., 1982). Ambos mapeos vienen dados por las ecuaciones

$$\begin{cases} \eta = q\,\theta \\ \xi = \log_a \dfrac{\rho}{\rho_0} - \rho \end{cases} \qquad \begin{cases} \eta = q\,\theta_j \quad j \in [1,\ldots N_{ang}] \\ \xi = \log_a \dfrac{\rho_1}{\rho_2} \quad i \in [1,\ldots N_{cir}] \end{cases} \tag{2.3}$$

donde (ρ,θ) son las coordenadas polares y (ξ,η) son las coordenadas log-polar. El valor ρ_0 es el radio de la circunferencia que separa ambos mapeos. Los valores q y a son constantes que vendrán determinadas por el tamaño y forma del sensor, en aquel primer caso de tipo CCD. En concreto, 1/q determina la mínima resolución angular del layout log-polar. La Figura 2.6 muestra cómo quedaría el modelo.

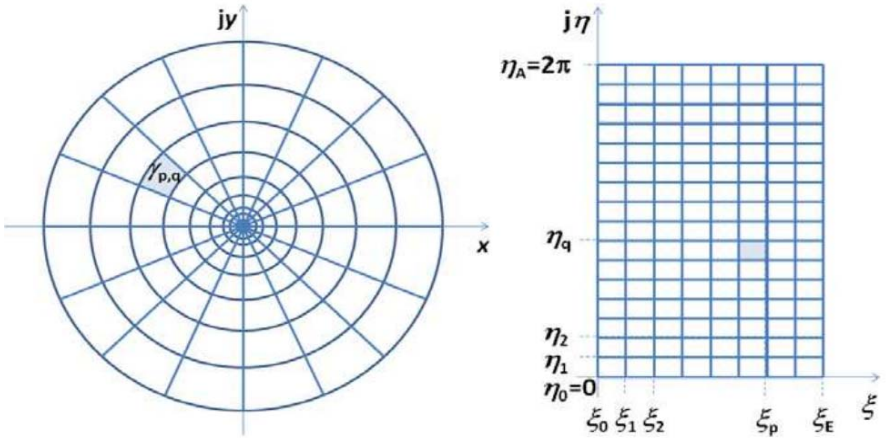


Figura 2-5: El modelo log(z) para el mapeo retino-cortical. En el centro de la representación de la izquierda se muestra un pequeño círculo, que queda fuera del modelo para evitar el problema de singularidad de la transformación (Traver y Bernardino, 2009). El plano retinal de la izquierda se mapea en el plano cortical de la derecha usando w=log(z). Circunferencias concéntricas y líneas radiales en el plano de la retina se transforman en líneas rectas en el plano cortical.

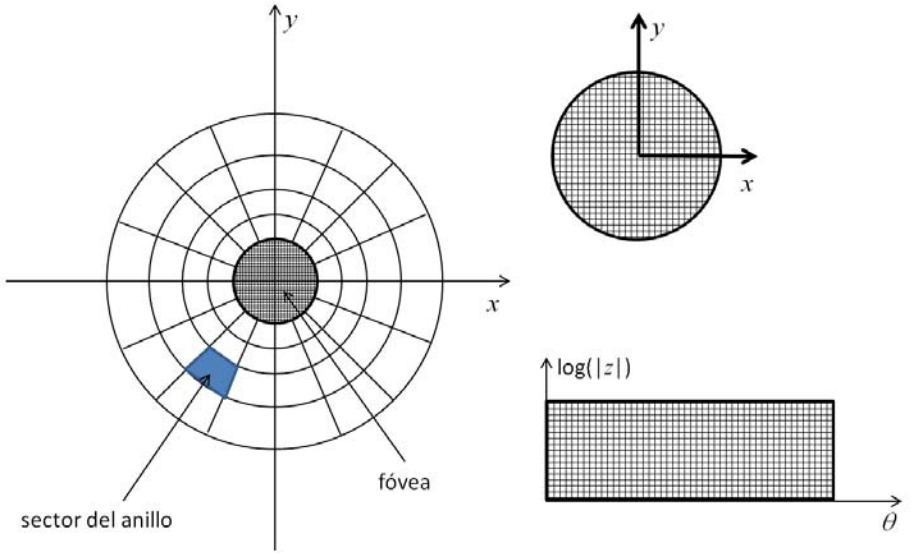


Figura 2-6: Transformación log-polar basada en dos mapeos distintos para la fóvea (uniforme) y la periferia (log-polar). Las imágenes de la derecha muestran la matriz de la fóvea en el plano cortical (arriba) y el de la periferia (abajo) (Bolduc y Levine, 1998)

En resumen, aunque la propuesta de usar dos modelos permitió en su momento la fabricación de un sensor real, la discontinuidad entre fóvea y periferia es una importante desventaja, que obliga a trabajar con dos mapas corticales distintos. Por ello, en paralelo al trabajo mencionado, Schwartz propuso eliminar el problema de la singularidad en el origen modificando la función que controla el mapeo, que sustituyó por la función $\log(z + a)$. Además, en sus trabajos de campo demostró que, seleccionando un valor adecuado para la constante a, era posible obtener un modelo aún más cercano al de primates o gatos que con la función $\log(z)$ (Schwartz, 1980).

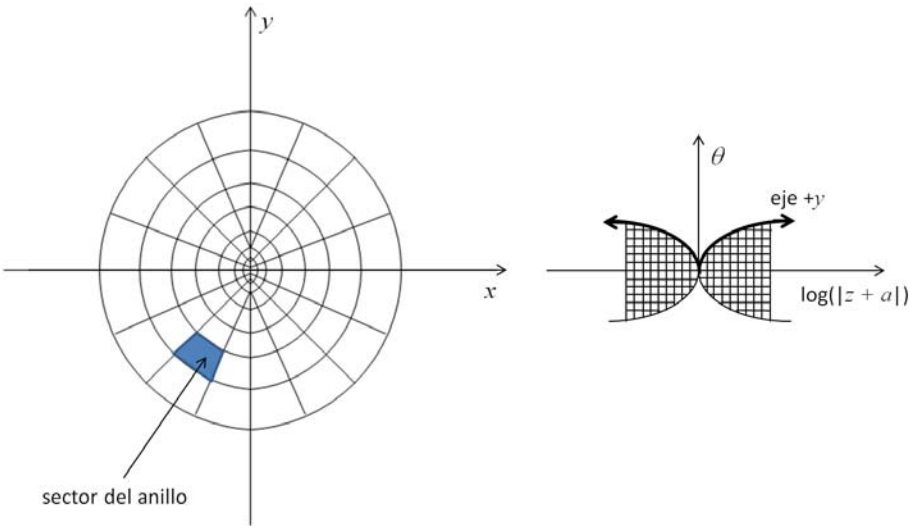


Figura 2-7: Transformación log-polar basada en la función $\log(z + a)$. La imagen de la izquierda muestra que el patrón es simétrico y elimina la singularidad en el origen, pero a costa de presentar sectores de anillos distintos en el eje y. La imagen de la derecha muestra la matriz en el plano cortical, con la característica forma de mariposa. En ella se ha marcado la posición del eje y (Bolduc y Levine, 1998)

Como muestra la Figura 2.7 la matriz en el plano cortical no presenta una forma rectangular. El lado derecho se mapea usando $\omega = \log(z + a)$, dando lugar a la parte derecha de la representación en el plano cortical, mientras que el izquierdo lo hace con la expresión $\bar{\omega} = 2 \cdot \log(a) - \log(-\bar{z} + a)$, dando lugar a la parte izquierda de la matriz cortical. El mapeo en sí es conforme dentro de cada mitad del plano. Sin embargo, en un sentido matemáticamente estricto, las propiedades de invarianza en escala y rotación, que sí presenta la transformación $\log(z)$, se pierden[5]. Además, como el patrón se 'pliega' en el eje

5 Sin embargo, si se considera que $|z| >> a$, entonces $\log(z + a) \approx \log(z)$, y se puede considerar que las propiedades de invarianza se mantienen

y (ver Figura 2.7), los círculos concéntricos y centrados en la fóvea, o las líneas rectas que pasan por ésta, ya no mapean a líneas rectas en el plano cortical.

2.2.2. Transformación cartesiana exponencial

El modelo cartesiano-exponencial aparece, a finales de los 80s, en las propuestas de César Bandera y Peter Scott (Bandera y Scott, 1989; Scott y Bandera, 1990; Bandera, 1994). Al igual que en la geometría log-polar, en este modelo la resolución disminuye al alejarnos del centro pero, a diferencia de la anterior, el campo receptivo de los *ring sectors* (o *region pixels* - *rexels*) es ahora cuadrado. Además, al alejarnos de la fóvea, el tamaño de éstos no cambia de forma continua, sino en saltos discretos, de forma que la relación de tamaño entre rexels de anillos consecutivos es una potencia de dos. En la Figura 2.8 se muestra una geometría cartesiana exponencial con una región periférica de tres anillos y fóvea. Se aprecia, sin embargo, que una línea que cruce desde el centro hasta el borde exterior de la cuadrícula atravesaría seis rexels. Cada anillo presenta un conjunto de subanillos de igual resolución. Así, la geometría de la Figura 2.8 se caracteriza por presentar tres anillos (m = 3) y dos subanillos (d = 2).

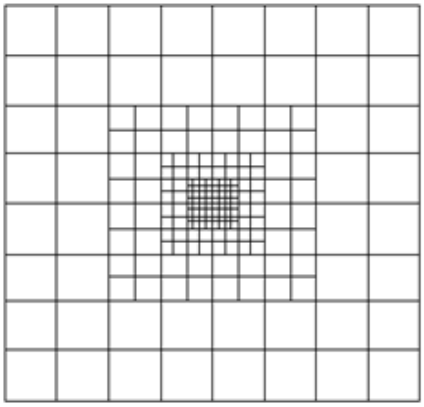


Figura 2-8: Geometría cartesiana exponencial con m = 3 y d = 2

La geometría cartesiana no presenta discontinuidad entre la fóvea y la periferia, para lo que la fóvea debe presentar un tamaño de $4d$ x $4d$. Además, al igual que la transformación log-polar, puede mapearse en la corteza asegurando que, líneas que pasen por la fóvea y cuadrados (no circunferencias) centrados en el origen, sean líneas rectas en el mapa cortical. La diferencia con respecto

al mapeo log-polar es que ahora el plano cortical no presenta un espaciado uniforme. Como muestra la Figura 2.9, al dividirse los anillos en porciones de tamaño prefijado, cada uno de ellos se asocia ahora a un arco de ángulo distinto.

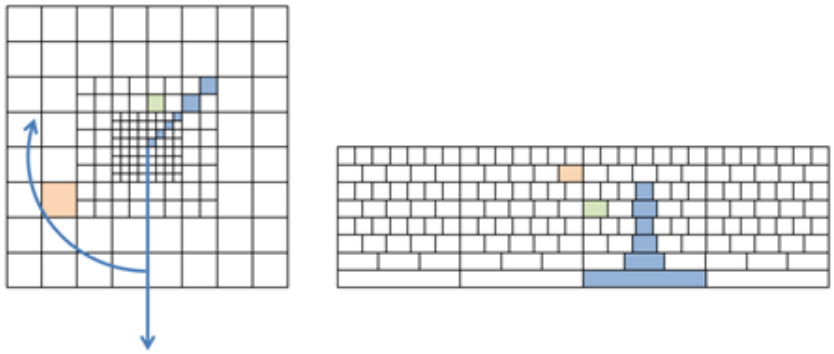


Figura 2-9: (izquierda) Imagen retinal en el patrón cartesiano-exponencial; y (derecha) correspondiente imagen en el plano cortical. En dicho plano cada rexel tiene asociado un arco de distinto tamaño. La línea marcada en azul en el plano de la retina se asocia a una línea en el plano cortical, pero esta línea no tiene un ancho constante.

La geometría cartesiano-exponencial comparte con la geometría log-polar la invarianza en rotación y escalado: cuando el patrón percibido en el plano de la retina gira o cambia en escala, su correspondiente proyección en el plano de la corteza se traslada en uno de los ejes (ver Figura 2.10).

Esta propiedad, empleada en el caso de la transformación log-polar pues se persigue trabajar en el plano cortical, organizado en ese caso como una matriz cartesiana uniforme (Sandini y Metta, 2002), no ha sido sin embargo empleada en el equivalente cartesiano-exponencial. Como presentamos en la Sección 2.3 de este Capítulo, la retinotopología cartesiano-exponencial busca hacer uso de la regularidad en el plano retinal para trabajar con estructuras jerárquicas (Bandera, 1994).

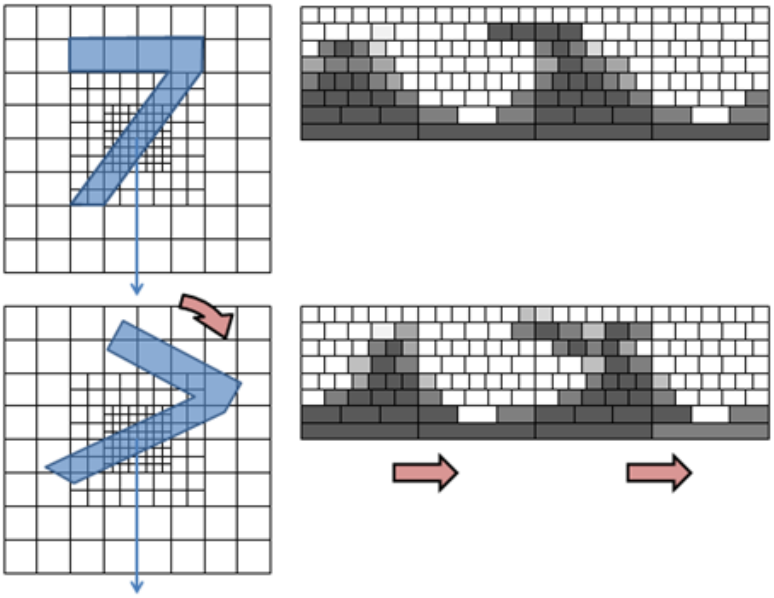


Figura 2-10: Invarianza a rotación: cuando el patrón percibido en la retina rota, su correspondiente proyección en el plano cortical se traslada (la distorsión que se aprecia en la imagen se debe al proceso de cuantificación, como ocurre en el caso log-polar (Traver y Bernardino, 2010))

2.2.2.1. Geometrías cartesiano-exponencial de fóvea desplazable

Como se describió en la Sección 2.1, al analizar los mecanismos que ha desarrollado nuestro cuerpo para mover el globo ocular, la emulación del control de la mirada humano no es obvio de replicar en los sistemas artificiales y, cuando se ha hecho, supone el diseño de un controlador complejo. Para evitar los pequeños desplazamientos del sensor, la fóvea podría desplazarse por la imagen, limitando el movimiento a aquellas situaciones en las que el objeto de interés se sale del campo de visión global de la cámara. Este proceso, no implementado en el caso de la geometría log-polar, ha dado lugar en la cartesiano-exponencial a las geometrías de fóvea desplazable (Camacho et al, 1996; Arrebola et al, 1996). Existen hasta cuatro variantes de Geometrías Multirresolución de Fóvea Desplazable (GMFD) (Arrebola et al, 1998; Coslado, 2004):

- *GMFD Básica*. Cada nivel de resolución se desplaza respecto a su posición centrada en función de dos constantes, s_v y s_h. Así, en la Figura 2.11(izquierda) no hay desplazamiento entre niveles en el eje horizontal (s_h = 0), y de valor uno en el vertical (s_v = 1)..

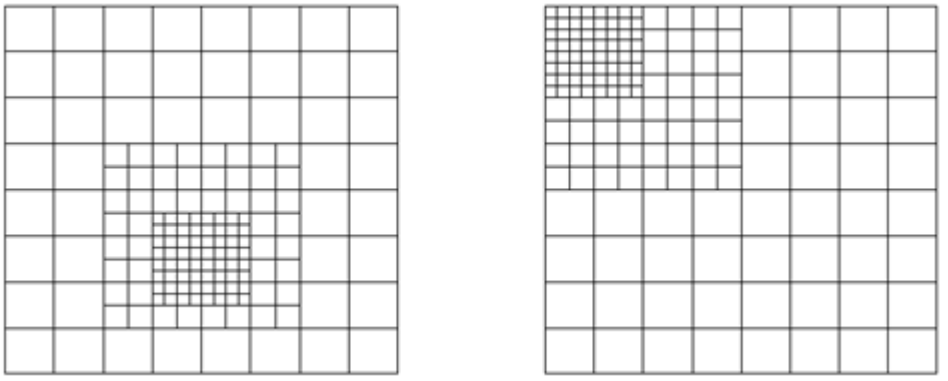


Figura 2-11: GMFD Básica: (izquierda) con s_h = 0 y s_v = 1; y (derecha) con s_h = -2 y s_v = -2

- *GMFD de movimiento generalizado*. Con el objetivo de ubicar la fóvea en cualquier zona del campo de visión, en la GMFD generalizada (Arrebola et al, 1997), cada nivel de resolución se desplaza ahora respecto a su posición centrada en función de dos vectores de constantes, $s_v^{(i)}$ y $s_h^{(i)}$. La Figura 2.12 muestra un par de ejemplos.

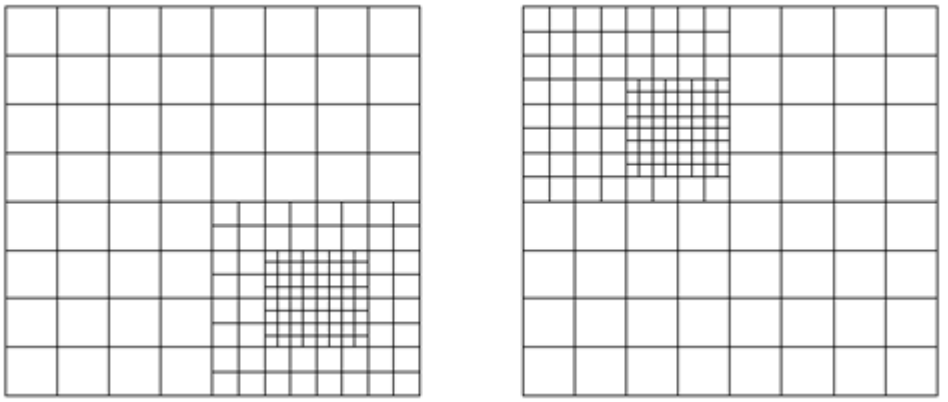


Figura 2-12: GMFD de movimiento generalizado: (izquierda) con s_h = {0,2} y s_v = {0,2}; y (derecha) con s_h = {2,-2} y s_v = {1,-2}

- *GMFD de fóvea de tamaño adaptativo*. En las geometrías anteriormente descritas, el tamaño de la fóvea o de los anillos permanece constante, como en las geometrías de fóvea centrada. Para adaptar la fóvea al tamaño de la región de interés, Pelegrín Camacho y Fabián Arrebola propusieron una estrategia para adaptar el tamaño de la fóvea y los anillos (Camacho et al., 1997a). Paralelamente, la

geometría admite desplazamientos constantes entre los niveles de resolución. La geometría se caracteriza ahora por cuatro parámetros, L_d, R_d, T_d y B_d, que almacenan el factor de subdivisión izquierdo, derecho, superior e inferior, respectivamente, de cada anillo de resolución. La Figura 2.13 muestra un par de ejemplos.

Al ser el desplazamiento entre anillos sucesivos constante, los límites de la fóvea o del resto de anillos sólo se pueden ubicar en determinadas posiciones del campo de visión.

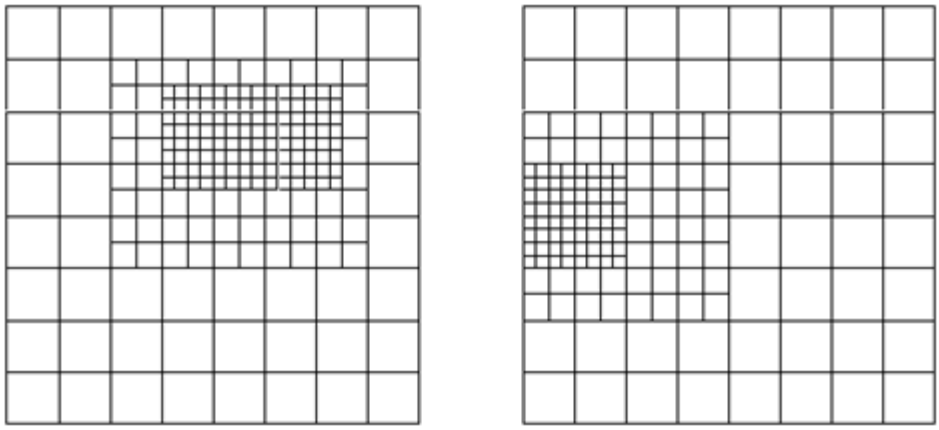


Figura 2-13: GMFD de movimiento adaptativo: (izquierda) con L_d = 2, R_d = 1, T_d = 1 y B_d = 3; y (derecha) con L_d = 0, R_d = 4, T_d = 2 y B_d = 2

- *GMFD de fóvea de tamaño optimizado.* La generalización del movimiento de la fóvea en el campo de visión puede extenderse al caso de fóvea adaptativa (Arrebola, 1998). La geometría se caracteriza ahora por cuatro vectores, los referidos a las dimensiones en ancho y alto de cada anillo de resolución, incluida la fóvea (h y v) y los referidos al desplazamiento (Sh y Sv). En la Figura 2.14 se muestran dos ejemplos. Los vectores aportan datos sobre la fóvea y el primer anillo, en ese orden.

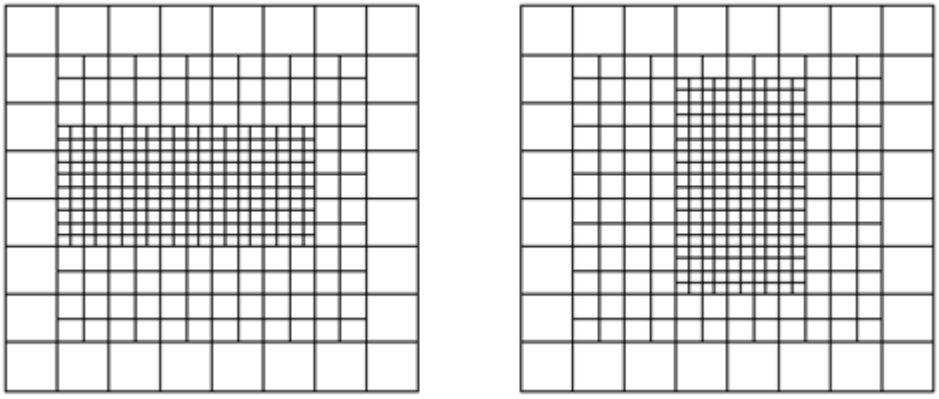


Figura 2-14: GMFD de movimiento optimizado: (izquierda) con h = {20,12}, v = {10,12}, Sh = {0,1} y Sv = {3,1}; y (derecha) con h = {10,12}, v = {18,12}, Sh = {4,1} y Sv = {1,1}

Las últimas implementaciones de geometría de fóvea desplazada permiten optimizar los recursos computacionales y de memoria, al englobar con precisión la región de interés en la fóvea, y al permitir obviar la complejidad del movimiento mecánico cuando el desplazamiento de la fóvea es pequeño y puede llevarse a cabo sin mover el sensor completo. Además, la obtención de la geometría desde un sensor uniforme puede hacerse en tiempo real usando implementaciones hardware, como muestran las implementaciones sobre la FPGA EPF10K-50 de Altera (Camacho et al., 2000) o sobre una FPGA Spartan II de Xilinx (González et al, 2002).

2.2.3. Otras aproximaciones

Trabajando de forma similar a las geometrías log-polar o cartesiana, Frederique Robert y Eric Dinet propusieron replicar la forma real de los conos y emplear una división del espacio en hexágonos (Robert y Dinet, 1999). Como se muestra en la Figura 2.15, la posición de cada hexágono viene dado por sus coordenadas polares y caracterizado por un radio, que viene determinado por la distancia al origen. En general, el método no es tan popular como los ya descritos, aunque en un estudio relativamente reciente se justifique el uso de este modelo por preservar mejor la información de la imagen uniforme (Robert-Inacio e Yushchenko, 2014).

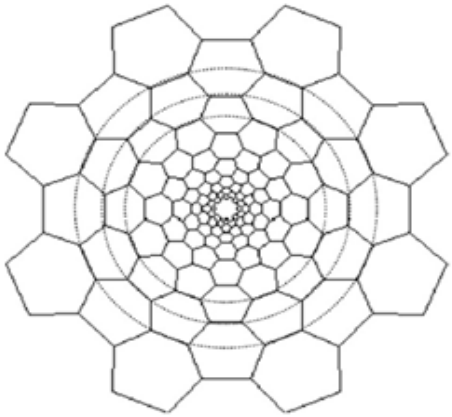


Figura 2-15: Geometría hexagonal

Fuera de las geometría log-polar o cartesiano-exponencial es posible también encontrar otras alternativas de muestreo variante en el espacio. La RWT (*Reciprocal Wedge Transform*) propuesta por Frank Tong y Ze-Nian Li preserva la linealidad en las características y ha sido empleada en tareas de seguimiento de líneas en carretera o estimación de profundidad (Tong y Li, 1995; Tong, 1995). Básicamente, la RWT define un mapeo entre los píxeles de la imagen uniforme desde el plano x-y al un nuevo plano u-v, en el que $u = 1/x$ y $v = y/x$. En la Figura 2.16 se muestra el resultado de la transformación. La reconstrucción o resultado de la derecha muestra el efecto de suavizado en los bordes, asociado a una reducción del volumen de datos.

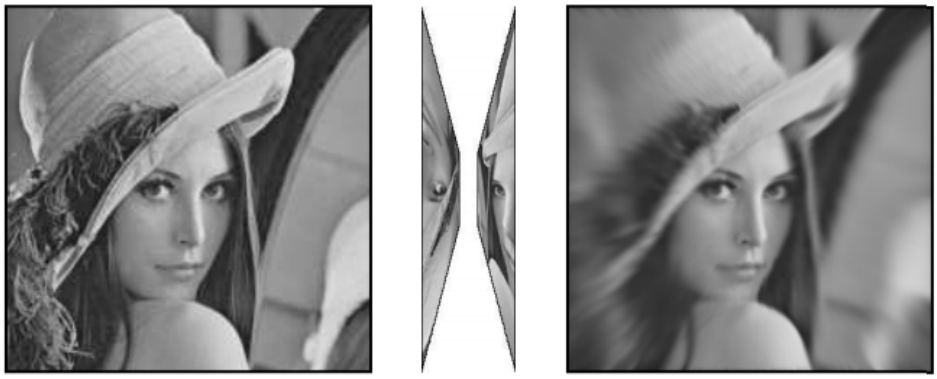


Figura 2-16: (izquierda) imagen de entrada; (centro) la imagen RWT muestra dos wedges; y (derecha) la imagen transformada de nuevo al dominio Cartesiano.

La transformación exponencial dimensionalmente independiente (DIEM, *Dimensionally-Independent Exponential Mapping*) mapea de forma distinta a lo largo de los ejes vertical u horizontal (Peters y Arcot, 1998). La idea es simple: las muestras de la imagen uniforme que formarán parte de la retinotopología variante en el espacio se toman de la intersección de líneas horizontales y verticales. Como muestran los ejemplos de la Figura 2.17, la principal ventaja del DIEM es su flexibilidad, al permitir que los parámetros se puedan elegir para cada eje de forma independiente. Así, por ejemplo, es posible muestrear con más detalle cualquier parte del campo de visión. En cierta forma, estas ventajas las comparte con las GMFD optimizadas.

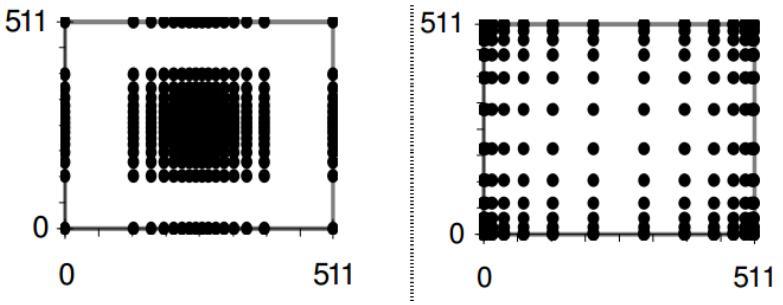


Figura 2-17: DIEM: Ejemplos de muestreos variantes en el espacio

2.2.4. Sensores no uniformes

En paralelo a la aparición de las propuestas biológicas y sus correspondientes desarrollos en el campo de las Ciencias de la Computación, la década de los 90s verá la aparición de sensores físicos cuyo diseño estará guiado por el paradigma de varianza en el espacio. Estos sensores buscan replicar, normalmente, la geometría de la transformación log-polar.

El diseño de un sensor específico de geometría no uniforme implica, en primer lugar, elegir los dos parámetros que son condicionados por la tecnología: el tamaño mínimo de píxel y el tamaño máximo del sensor. Ambos determinarán, además, otro parámetro básico: el número total de píxeles del sensor. En el caso de las geometrías no uniformes, la relación entre este número y los dos parámetros citados no es directa, como sí lo es en el caso de un sensor uniforme. Para el caso de la geometría log-polar será además necesario fijar la relación entre los rexels de mayor y menor tamaño, que definiremos por la constante R (Sandini y Metta, 2002).

En la primera implementación que, usando tecnología *Charged Coupled Device* (CCD), se fabricó en el IMEC (*Interuniversitary MicroElectronics Center*) belga a propuesta de Giulio Sandini y su equipo (Van der Spiegel et al., 1989), el tamaño mínimo de píxel era de 30 μm x 30 μm, y el diámetro del sensor ocupó 9.4 mm. El sensor tenía 30 anillos, cada uno de los cuales disponía de 64 fotorreceptores. La fóvea se organizaba como una matriz regular de 102 píxeles (hay que recordar que éste era el esquema seguido por este grupo para evitar la singularidad de la transformación log-polar en el origen). En total el sensor cuenta con 2022 fotorreceptores. La Figura 2.18 (izquierda) muestra una imagen de dicho sensor, en la que se ha ampliado la zona de la fóvea y la transición con la región periférica. El valor de R estaba en 13.7, pues el rexel de mayor tamaño era de 412 μm x 412 μm. La tasa de adquisición estaba en los 50 fotogramas por segundo.

Un tercer parámetro importante es la relación entre el tamaño del sensor y el del menor píxel (constante Q). El valor de este parámetro determina el tamaño de una imagen de resolución uniforme que tendrá el mismo campo de visión y la misma máxima resolución que su correspondiente sensor log-polar. Así, para el CCD descrito, Q es igual a 300. Esto es, si queremos emplear un sensor uniforme para replicar este sensor log-polar, éste deberá tener 300 x 300 píxeles.

Las nuevas versiones de sensores log-polar usarán tecnología CMOS. En el marco del proyecto SVAVISCA[6], el grupo de Giulio Sandini fabricará en Tower (Israel) un sensor de tamaño mínimo de píxel de 7 μm, usando tecnología CMOS de 0.35 μm. El conjunto de fotorreceptores en la periferia será de 252 x 110 = 27720, mientras que la fóvea será de 5473 píxeles. Las constantes R y Q serán ahora de 17 y 1100 respectivamente. El diámetro del sensor es de 7,1 mm. En el IMEC se fabricará una versión color de este sensor, usando patrones similares al conocido Bayer usado en sensores uniformes (Figura 2.19). En la Figura 2.18 (centro) se muestra el aspecto de otro sensor CMOS, el diseñado y fabricado por Robert Wodnicki y su equipo. Al igual que las soluciones propuestas por Sandini, el sensor presenta una fóvea uniforme rodeada por anillos de tamaño de fotorreceptor creciente.

6 http://www.lira.dist.unige.it/projects/logpolar/press.html

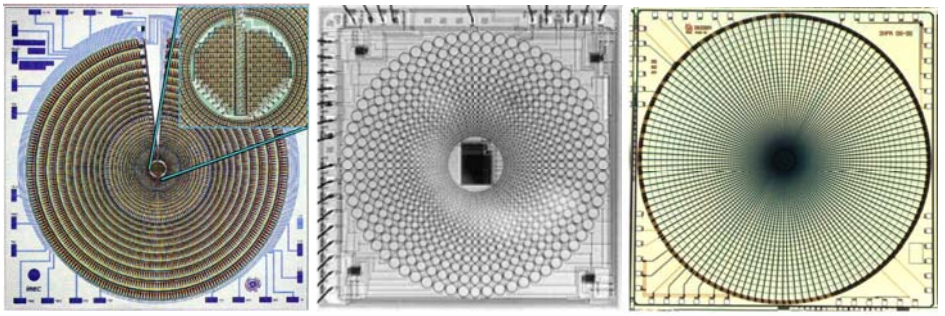


Figura 2-18: Sensores que imitan la retina: (izquierda) sensor CCD © 2002 IEEE (Van der Spiegel et al., 1989); (centro) sensor CMOS © 1995 IEEE (Wodnicki et al., 1995); y (derecha) sensor CMOS (Courtesy by Cypress Semiconductor Image Sensor) (Pardo et al., 1997)

Como muestra la Figura 2.18 (izquierda), en el sensor CCD se ha dejado de fabricar toda una fila de rexels para poder dejar espacio a las conexiones que acceden a los fotorreceptores de las zonas foveal y periférica (señales de control y reloj). En los siguientes diseños CMOS (Ferrari et al, 1995; Pardo et al., 1997), fabricados por el consorcio formado por el IMEC y el IBIDEM, estas señales se trazan a través de canales radiales (Figura 2.20).

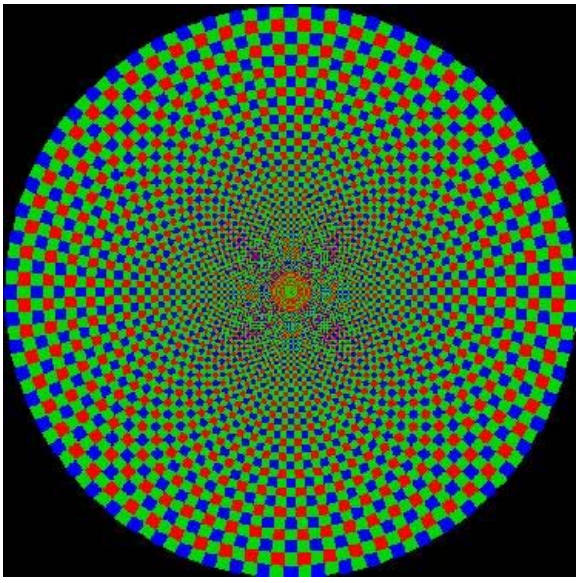


Figura 2-19: Sensor CMOS color desarrollado en el IMEC en el marco del proyecto europeo SVAVISCA

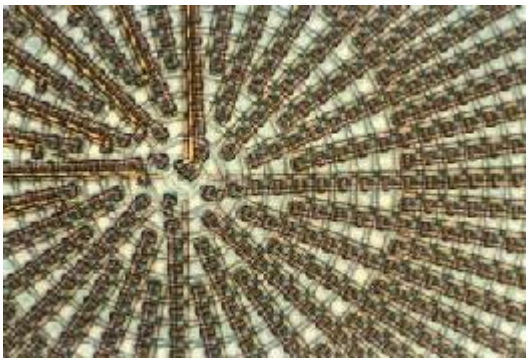


Figura 2-20: Región central del sensor CMOS FUGA18

Aunque menos citado, el IMEC, en colaboración con Corticon Inc. y NSF, también desarrolló por esta misma época un sensor de distribución cartesiana para aplicaciones de seguimiento (Etienne-Cummings et al., 2000; van der Spiegel et al., 2002). El sensor, mostrado en la Figura 2.21, consta de una fóvea y un único anillo de resolución, y está diseñado para mantener un objeto determinado en la fóvea. Mientras el objeto puede ser capturado por la fóvea, el sistema manda órdenes a los motores para que lo muevan y mantengan el objeto centrado. Cuando el objeto sale de la fóvea, la estimación de su nuevo centroide recae sobre el procesado de la región periférica. Todo el procesamiento requerido para ello está integrado en el hardware del sensor.

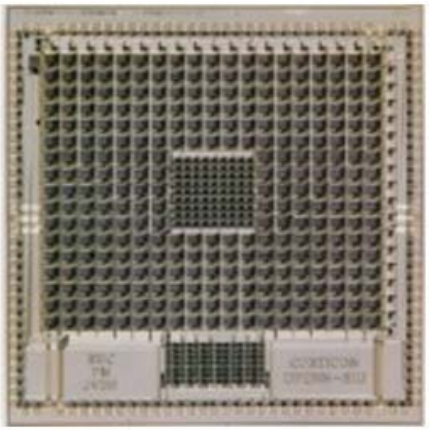


Figura 2-21: Microfotografía del chip para seguimiento desarrollado por Ralph Etienne-Cummings, Jan Van der Spiegel, Paul Mueller y Mao-zhu Zhang en Corticon Inc.

Finalmente comentar que, frente al desarrollo de sensores de geometría específica, las transformaciones log-polar o cartesiano-exponencial también se han obtenido usando dispositivos electrónicos dedicados que, en tiempo real, transforman la imagen uniforme del sensor para obtener la distribución deseada. En el caso de la geometría log-polar, estos trabajos son ligeramente anteriores al desarrollo de los sensores (Rojer y Schwartz, 1990; Weiman y Juday, 1990; Engel et al. 1994). En aquella época presentaban la ventaja de usar placas de desarrollo estándares, disponibles en el mercado a un precio razonable. Aunque como desventaja se esgrimió que estos sistemas estaban limitados por el tamaño del sensor uniforme (Sandini y Metta, 2002), es importante destacar que esta desventaja ya no existe hoy día, y es fácil encontrar sensores de 5MP a un precio muy razonable. Dada la evolución actual en dicho campo, es de prever que podamos emplear en nuestros prototipos, en los próximos años, sensores cercanos a los 20MP. En ese caso, no será difícil construir sensores de geometría no uniforme capturando los datos de los fotorreceptores ubicados en las posiciones que determine la transformación elegida.

2.3. La pirámide irregular foveal

Como se describió en la Sección 2.1, la estructura de la retina humana no sólo comprende el conjunto de fotorreceptores, sino también un conjunto de células que, organizadas jerárquicamente, trabajan con esa información para responder al contraste, al movimiento o al color. Cada retina contiene aproximadamente 100 millones de fotorreceptores, varias decenas de millones de neuronas horizontales, bipolares y amacrinas, y aproximadamente sólo un millón de neuronas ganglionares (ver Figura 2.22).

Sería pretencioso tratar de replicar esta estructura. Pero de ella podemos aprender varias lecciones. La primera y más importante es que la información visual debería ser tratada por un sistema realmente jerárquico: el cerebro no puede ser impresionado con toda la información capturada por los fotorreceptores pues difícilmente podrá trabajar con toda ella. Por otra parte, cada una de estas neuronas (ciertamente unas más que otras) comparte información con las neuronas de su nivel y la procesa y aporta al nivel superior. Este esquema, repetido en la corteza visual, será el germen de los algoritmos artificiales de procesamiento jerárquico que se presentan brevemente en el Apartado 2.3.1 y, en concreto, de la pirámide irregular foveal, estructura propuesta en esta Tesis y que se introduce en el Apartado 2.3.2.

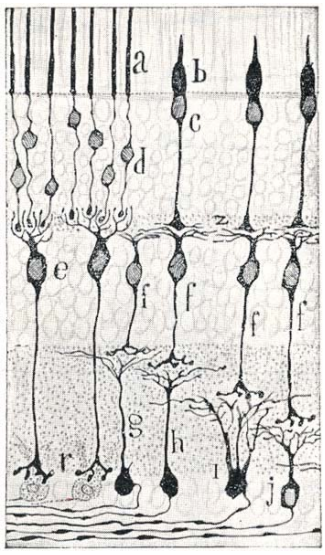


Figura 2-22: Esquema destinado a mostrar los cauces separados al través de la retina del impulso recogido por conos y bastoncitos de los mamíferos. — *a*, bastoncitos; *b*, conos; *e*, células bipolares para bastón; *f*, células bipolares para conos; *i*, *h*, *g*, *z*, células ganglionicas (Cajal, 1891)

(http://cvc.cervantes.es/ciencia/cajal/cajal_recuerdos/default.htm)

2.3.1. Algoritmos jerárquicos de procesamiento de imagen

El principio general que se persigue con el procesamiento jerárquico en general fue concisamente descrito por Jean M. Jolion y Montanvert para el caso de la pirámide: '*a global interpretation is obtained by a local evidence accumulation*' (Jolion y Montanvert, 1992). Para conseguir esta acumulación de evidencias locales, las estructuras jerárquicas representan el contenido de la imagen en distintos niveles de abstracción, que se pueden definir de forma general como grafos de nodos V_l conectados por un conjunto de arcos E_l. Estos arcos definen las relaciones horizontales en la estructura jerárquica y representan la vecindad de cada nodo en ese mismo nivel (arcos intra-nivel). Otro conjunto de arcos definen las relaciones verticales, conectando nodos entre niveles adyacentes de la pirámide (arcos inter-nivel). Este segundo conjunto de arcos establecen las relaciones de dependencia entre cada nodo del nivel l + 1 y un conjunto de nodos del nivel l (la denominada ventana de reducción). Comúnmente, los nodos que forman la ventana de reducción se denominan como los hijos del nodo que la define. El valor de este padre se calcula a partir de los valores de sus hijos usando una función de reducción, siendo el cociente entre el número de nodos del nivel l y el del nivel l + 1 el denominado factor de reducción. Usando este esquema, la acumulación de las evidencias locales se consigue con la sucesiva construcción del grafo G_{l+1}, asociado al nivel l + 1, desde el grafo asociado al nivel inferior G_l. Este proceso consta de tres pasos (Marfil et al., 2006):

1. Selección de los nodos del grafo G_{l+1} entre los nodos de V_l. Este proceso de selección es un diezmado, en el que los nodos V_{l+1} suelen nombrarse como nodos supervivientes.

2. *Establecimiento de los enlaces inter-nivel*. Cada nodo del grafo G_l se enlaza a un nodo en G_{l+1}, definiendo una partición en V_l.

3. Establecimiento de los enlaces intra-nivel. El conjunto de arcos E_{l+1} se obtiene definiendo las relaciones de adyacencia entre los nodos V_{l+1}.

Las relaciones padre-hijos definidas por la ventana de reducción se pueden extender hasta la base de la jerarquía. El conjunto de nodos de la base unidos a un nodo de cualquier de los niveles superiores forman su campo receptivo.

La eficiencia de una estructura jerárquica para resolver una determinada aplicación (segmentación en regiones, detección de contornos, etc.) vendrá influenciada por dos características: la estructura de datos y el esquema de diezmado, definidas respectivamente por los enlaces intra- e inter-nivel. La elección de una estructura de datos concreta determina la información que se puede codificar en cada nivel de la jerarquía. Define por tanto la red que forman los enlaces E_{l+1}, por lo que podemos decir que determinan las propiedades horizontales de la jerarquía. Por otra parte, el esquema de diezmado usado para construir cada nivel determinará la dinámica (altura, preservación de detalles, etc.) de la jerarquía reflejada en qué nodos sobreviven o cómo se enlazan los nodos de niveles consecutivos. Son las denominadas propiedades verticales. Teniendo en cuenta estas propiedades, las arquitecturas jerárquicas de procesamiento se han clasificado en regulares o irregulares.

En las jerarquías regulares la estructura de datos es rígida, con relaciones intra-nivel fijas y el factor de reducción constante. Sólo las relaciones inter-nivel se pueden cambiar para adaptar la estructura jerárquica al contenido de la imagen. En general presentan la ventaja de que, al trabajar con una estructura de nivel inicialmente conocida, ésta suele definirse como una imagen, con lo que el software desarrollado para trabajar sobre ellas suele ser muy eficiente. Además, los arcos intra-nivel son fijos (en la imagen, las vecindades de cada píxel) y pueden codificarse eficientemente (en la imagen podrán extraerse directamente de la propia posición (x,y) del píxel). Sin embargo esta inflexibilidad también tiene sus desventajas: campos receptivos desconectados, gran variación en los resultados cuando la imagen de entrada se desplaza levemente, o incapacidad para representar, bien como región o bien como borde, entidades alargadas. Las pirámides irregulares presentan una estructura de datos más flexible, en la que cada nivel se representa como un grafo cuyo número de nodos vendrá determinado por el proceso de diezmado. Las

relaciones inter- e intra-nivel son ahora definidas en el proceso de construcción de la estructura completa. En general, desde un punto de vista siempre software, son computacionalmente más costosas, pese a las soluciones que se han buscado para mejorar este aspecto.

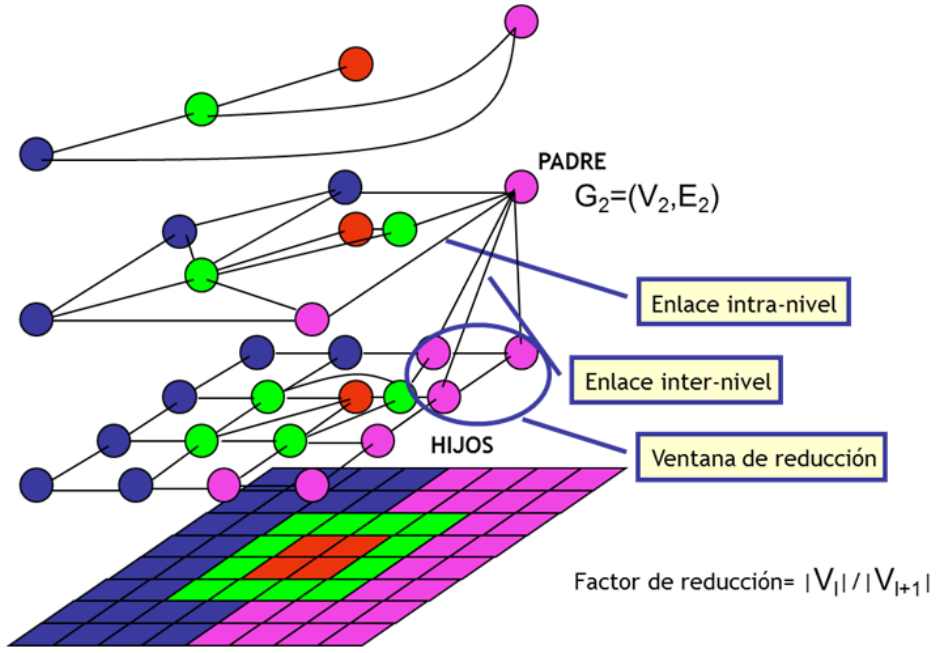


Figura 2-23: Notación en estructuras jerárquicas

2.3.2. Procesamiento jerárquico de imágenes multirresolución

Con frecuencia, las retinotopologías variantes en el espacio y las aproximaciones multirresolución se presentan como alternativas que compiten por la solución al problema del tratamiento de imágenes de gran tamaño. En parte, esta separación de las dos opciones surge del hecho de que no encontraremos fácilmente en la literatura la aplicación de un algoritmo jerárquico sobre la transformación log-polar, normalmente equiparada con muestreo variante en el espacio (podríamos citar el trabajo de Sumitha y Siebert (2006), en el que la jerarquía se emplea más para extraer características que para reorganizar el contenido de la imagen).

Sin embargo, ese no es el caso de las geometrías cartesiano-exponenciales. Así, ya en su trabajo de 1989, César Bandera planteaba organizar el contenido

del mapeo cartesiano-exponencial como una estructura jerárquica, el denominado polígono foveal, en contraposición a la entonces popular pirámide (Jolion y Montanvert, 1992; Marfil et al., 2006). La Figura 2.24 muestra esquemáticamente cómo se organizan ambas estructuras. Como se trata de reflejar en la figura, el polígono presenta un número más reducido de procesadores que la pirámide, justificado en la diferencia entre sensar un campo de visión con un esquema uniforme o un esquema variante en el espacio. Esto propició que, en su versión regular (la única conocida hasta 2014), se haya implementado en hardware desde mediados de la década de los 90s (Du et al., 1995; Du et al., 1996).

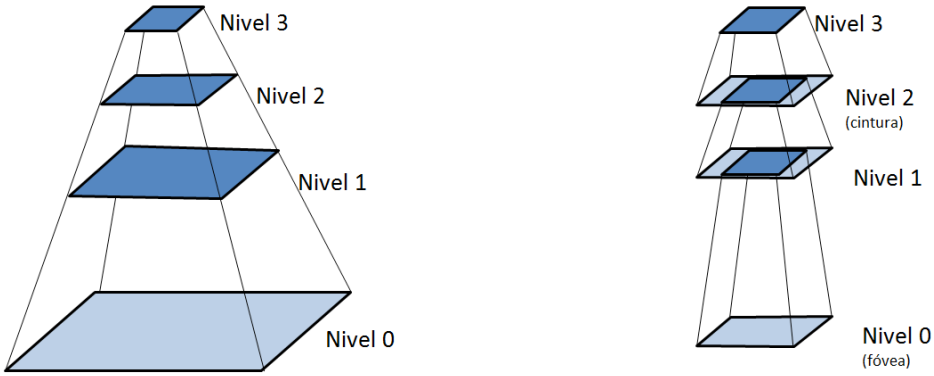


Figura 2-24: (izquierda) Esquema de la pirámide regular; y (derecha) el polígono foveal

El polígono foveal está formado por niveles de resolución uniforme, en el que la fóvea ocupa la base o nivel inferior, y el resto de niveles combinan información que proviene del diezmado del nivel inferior (la zona más oscura en los niveles del polígono de la Figura 2.24 (derecha)) con rexeles de un determinado anillo de la geometría cartesiano-exponencial. En la cintura del polígono ya está representado todo el campo visual, al unirse el anillo más externo de la geometría cartesiano-exponencial al resultado del diezmado del nivel inferior. Sobre la cintura se construye una pirámide, que ahora trabaja ya con la información de todo el campo visual. El escenario en sí explica el distinto planteamiento con el que nacen geometrías log-polar y cartesiano-exponencial. En la primera se busca trabajar en el plano cortical, mientras que en la segunda la idea es más la de disponer de distintos niveles de abstracción con los que poder resolver distintas tareas (si se pretende reconocer objetos se trabajará con la fóvea, pero si se busca detectar movimiento (o flujo óptico como será el caso original de César Bandera) se trabaja ya sobre la cintura del polígono o, incluso, sobre niveles construidos sobre ésta).

Normalmente, el proceso de diezmado dentro del polígono foveal se ha llevado a cabo usando aproximaciones regulares. En aplicaciones de segmentación de imagen basadas en el nivel de gris, dicho diezmado se ha basado en la estrategia propuesta por Peter Burt en 1981 para la pirámide (*Linked pyramid*), adaptado para su empleo en el marco del polígono foveal (Arrebola et al., 1997; Coslado, 2004). En este proceso de diezmado, cada nivel del polígono foveal se puede considerar como una imagen que, a su vez, se diezma para formar una nueva imagen de menor tamaño o resolución. Se le añade entonces el nivel de resolución correspondiente, construyendo una imagen que puede ser de nuevo procesada. En parte, la obligación de tener que representar cada nivel como una imagen justifica la existencia de los niveles que aparecen bajo la cintura del polígono, niveles que, sin embargo, no se han llegado a emplear como entrada para ninguna tarea en especial. Los problemas que esta estrategia de diezmado presenta fueron publicados poco después de su propuesta por Peter Burt (Antonisse, 1982; Bister et al., 1990), y su uso sólo se justifica por la simplicidad que el manejo de imágenes, en las que la conectividad viene supuesta, puede aportar (Traver and Bernardino, 2010). Empleamos el término supuesta pues, si bien este esquema considera que dos píxeles vecinos en la imagen están realmente conectados en la imagen original, esto no tiene porqué ser realmente cierto. Como se ha comentado anteriormente, la superación de este error vendrá de la mano del empleo de las denominadas pirámides irregulares (Marfil et al., 2006). En el marco del polígono foveal, sólo recientemente se ha empleado una estrategia irregular para segmentar imágenes en color, en el que el proceso de diezmado se ha llevado a cabo usando la estrategia de la Pirámide Irregular Acotada (*Bounded Irregular Pyramid*, BIP) (Marfil et al., 2014). El esquema implica que las zonas internas a los anillos en la Figura 2.24(derecha) (así como la cintura del polígono y todos los niveles construidos sobre ésta) serán realmente grafos y no imágenes. Esto complica las estructuras empleadas, pero evita los problemas que presentan las aproximaciones regulares.

Por tanto, podemos resumir que, como combinación de procesado jerárquico e imagen multirresolución, el polígono foveal constituye una propuesta interesante, justificada principalmente en el marco de un procesado regular, en el que cada nivel de la estructura es una imagen. Sin embargo, la estructura en sí también tiene desventajas. ¿Cómo trabajar sobre un campo visual realmente multirresolución? En el trabajo de José Martínez y Leopoldo Altamirano (2006) se documentan las dificultades que presenta trabajar sobre imágenes que sólo parcialmente cubren el campo de visión cuando se pretende implementar un algoritmo de seguimiento. Ejecutar dicha tarea sobre la cintura del polígono supone trabajar con una fóvea de baja resolución, tan reducida como la del

anillo más externo de la periferia. En verdad, y contrario a lo que encontramos en la retina humana, los datos de la fóvea sufren en el polígono foveal un continuo proceso de diezmado. Como se ha ya comentado, al final los algoritmos, como ya ocurre en la primera propuesta de César Bandera, trabajan sobre una imagen de resolución uniforme, sea la cintura o un nivel superior, en la que los datos que vienen de la fóvea sólo han sido beneficiados en el sentido en que su diezmado está más dirigido por ellos mismos que los de los anillos, que provienen, en las retinas emuladas por software, de simples promediados.

Para evitar este problema y trabajar sobre datos multirresolución, José Martínez y Leopoldo Altamirano proponen una nueva forma de emular una retina artificial desde un sensor de resolución uniforme, construyendo una imagen cuyos píxeles se capturan de distintas posiciones del mallado uniforme (Figura 2.25).

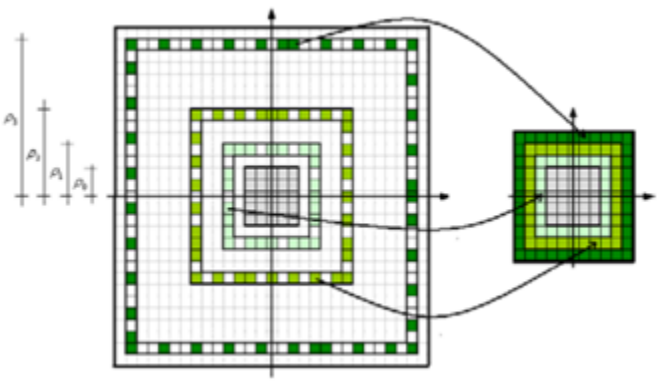


Figura 2-25: Geometría cartesiana propuesta por José Martínez y Leopoldo Altamirano (2006)

La propuesta se basa en el mallado pseudo log-polar (PPG) (Averbuch et al., 2001), aunque la formulación se ha generalizado para permitir el movimiento de la fóvea por toda la imagen.

La pirámide irregular foveal que se propone en esta Tesis combina las ideas de trabajar con un imagen realmente multirresolución, que incluye el campo de visión completo que abarcan sensor y óptica, y el procesamiento jerárquico propio de las pirámides irregulares. En lugar de montar una imagen como se muestra en la Figura 2.25, la imagen multirresolución será realmente un grafo pues, como se ha descrito en el Apartado 2.3.1, la flexibilidad de las estructuras irregulares no puede encorsetarse en el mallado uniforme de la imagen.

2.3.3. La pirámide irregular foveal

En la Figura 2.26 se esquematiza la estructura de la pirámide irregular foveal. En realidad, la única diferencia con una pirámide irregular se manifiesta en la base de la misma, que ahora está definida como una imagen multirresolución, con una configuración que, en esta Tesis, seguirá el patrón de la GMFD de fóvea de tamaño adaptativo. Representada como un grafo, la base puede ser procesada por cualquier estrategia de diezmado propuesta en el marco de las pirámides irregulares, con la única salvedad de que sean capaces de trabajar con grafos simples. Lo importante es que dicha base presenta una mayor densidad de nodos en la zona de la fóvea, que decrece conforme nos alejamos de ésta. Aunque la figura sólo es un esquema, se ha tratado de reflejar en ella una importante propiedad de la pirámide irregular foveal: el hecho de que, en los distintos niveles de la pirámide, la densidad de nodos asociados, por enlazado inter-nivel, a la fóvea será siempre mayor, por lo que los distintos niveles siempre proporcionarán una visión más detallada de la fóvea. Conforme se suba en la estructura esta característica se irá difuminando, y en los niveles superiores, en los que el nivel de abstracción de la escena es alto, el nivel de detalle será siempre bajo, pero siempre se habrán conservado mejor los detalles de la fóvea que los de la zona periférica. La estructura evita así los problemas del polígono foveal, combinando representación multirresolución y jerarquía de forma más íntima, pues ahora todos los niveles de abstracción de la jerarquía representan la escena completa a distintas resoluciones.

Los siguientes Capítulos de esta Tesis detallan el trabajo realizado para implementar esta estructura en un AP SoC. La implementación de un algoritmo de diezmado que pueda sintetizarse eficientemente en el AP SoC se analiza en el Capítulo 3. En concreto, el algoritmo será capaz de segmentar la imagen en regiones de color uniforme. Con el objeto de embeber en la estructura un sistema de control del movimiento de la fóvea, el Capítulo 3 también describe la implementación de un mecanismo de atención, que usando determinadas características de las regiones (contraste color e intensidad, simetría,…) guiará la fóvea a aquella que sea considerada como más relevante. La adquisición de la base de la pirámide irregular foveal, que responde a la formulación propia de las GMFD de fóvea de tamaño adaptativo, se analiza en el Capítulo 4, en el que también se describe los aspectos de implementación finales del sistema completo.

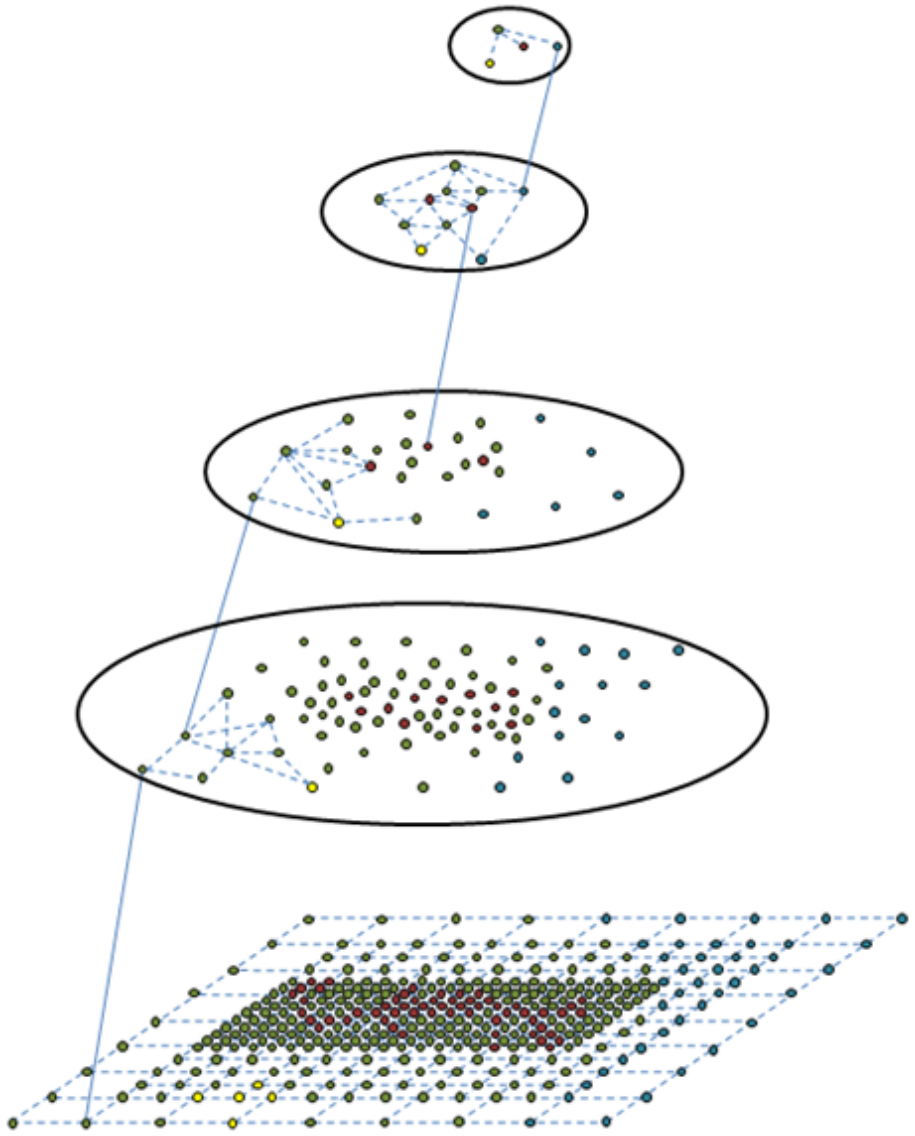


Figura 2-26: Representación muy esquematizada de la pirámide irregular foveal. En la figura se muestran algunos enlaces intra-nivel (completo por ejemplo en la base si se considera vecindad-4) y la supervivencia de un nodo por nivel (por claridad no se muestran todos estos enlaces). La estructura representada constaría de cinco niveles, incluida la base. Cualquier algoritmo de diezmado irregular (estocástico, D3P, BIP…) potenciará la supervivencia de una mayor densidad de nodos sobre la zona de la fóvea, pues en ésta habrá siempre no sólo mayor densidad de nodos, sino también mayor información sobre bordes o detalles.

2.4. Discusión

El diseño de un sistema de visión artificial que imite en detalle el funcionamiento del sistema visual humano es una utopía difícilmente realizable y que, en ocasiones, puede llegar a convertirse en una experiencia poco agradable. Como se ha introducido muy brevemente en este Capítulo, sólo la retina es ya una estructura de preprocesado jerárquica que combina numerosos tipos de células, en la que los fotorreceptores, bien conos o bastones, se reparten de forma no uniforme, pero en la que otras células, como las horizontales y las bipolares para la detección de bordes (Park et al., 2003), también lo hacen. La existencia de un sensor no uniforme permite cubrir un amplio campo de visión en el que sólo una pequeña porción se captura a máxima resolución. El resultado es poder procesar toda la información en tiempo real, pese al tamaño del campo de visión. En la década de los 90s, esta afirmación sirvió de base para la aparición de una firme línea de trabajo que, mayoritariamente apoyada en la transformación log-polar, postuló que sólo con un esquema de captura variante en el espacio sería posible construir robots con capacidades de percepción visual similares a las nuestras. Con los años, dicha línea ha perdido vigencia. Por un lado, la imposibilidad de trabajar con imágenes uniformes ha sido resuelta por el incremento en la capacidad de cómputo de los ordenadores personales. Por otra parte debe también resaltarse que su quizás mayor competidor, el procesamiento multi-escala o multirresolución, ha ganado un mayor peso en la comunidad científica.

Paralelamente, el sistema de visión de resolución variante en el espacio difícilmente se entiende sin su aplicación en el marco más amplio de la visión activa. Poder mover los sensores con precisión a una gran velocidad tiene connotaciones muy fuertes con los sistemas de control de la mirada o los mecanismos de atención pero, principalmente, con el control mecánico que ello implica. Los trabajos en controladores no-lineales que imiten el resultado de los sistemas de movimiento del globo ocular en las personas, u otros seres vivos avanzados, no son numerosos y aportan complejas soluciones. Normalmente sin embargo no se han discutido otras aproximaciones al problema: si se debe mover la fóvea para englobar al objeto de interés, habrá que mover todo el sensor.

Este Capítulo ha asentado, de forma muy rápida y particularizada para el ser humano, la base de las soluciones planteadas por la naturaleza ante el problema de sensorizar un amplio campo de visión usando una retina en la que solo se deben ubicar un número procesable de fotorreceptores. En resumen, las dos soluciones (sensor no uniforme y un preciso control de movimiento) se trataron de imitar en los sistemas de visión activa variantes en el espacio. Con

la aparición en estos últimos años de sensores económicamente asequibles y de tamaño cada vez mayor, la opción de emplearlos como base para conseguir sensores no uniformes cobra nuevo interés. Si nuestro sistema de procesamiento de imagen puede trabajar relajadamente con un volumen aproximado de medio megapíxeles, podemos repartir un diez por ciento de estos píxeles en la periferia de un perfil no uniforme y dejar una fóvea de casi este mismo tamaño. El resultado, para el que podríamos partir de un sensor de 5 Mpíxeles tendría el campo de visión de éste, pero una fóvea que permitiría percibir en detalle prácticamente medio megapíxel.

Empleando las geometrías multirresolución de fóvea adaptativa y desplazable es posible además combinar algorítmicamente el empleo de un muestreo variante en el espacio con el procesamiento multi-escala. Ambas estrategias han sido analizadas en este Capítulo, en un estudio que, partiendo de un repaso superficial del estado del arte, ha tratado de profundizar en aquellas direcciones más relacionadas con la arquitectura finalmente implementada en esta Tesis Doctoral, la denominada Pirámide Irregular Foveal. Finalmente, el empleo de una geometría de fóvea desplazable construida sobre un sensor de distribución uniforme de fotorreceptores plantea una duda razonable sobre la motorización final del sensor de captura, ¿por qué no combinar los movimientos mecánicos con el desplazamiento de la fóvea dentro de la retina? Descartar la opción simplemente porque no es biológicamente plausible puede no ser la más práctica desde el punto de vista de la implementación software.

Capítulo 3

Segmentación y atención

Como se analizó en el Capítulo 2, el sistema de visión humano presenta un conjunto de características, relacionadas con la adaptabilidad y robustez, que le permiten analizar y procesar la información visual de una escena compleja de una forma sumamente eficiente. Los trabajos en psicología y fisiología demuestran que dicha eficiencia sienta sus bases en la atención visual, el proceso por el cual se filtra la información irrelevante y se limita el procesamiento a aquellos elementos que son considerados realmente importantes (Duncan, 1984). Como se refirió en el Capítulo 2 (Figura 2.1), cuando una persona observa una escena no lo hace como un todo, sino que lleva a cabo una serie de fijaciones visuales en posiciones relevantes de la misma (Martínez-Conde et al., 2004). Como ya se mencionó también en dicho Capítulo, el sistema motor que rodea al ojo le permite a éste realizar estas fijaciones usando movimientos rápidos y muy precisos, con los que logra encuadrar estas regiones relevantes en la fóvea de la retina. Las fijaciones obligan, además, a que ciertas zonas del campo visual se localicen en la periferia, y ello hace que sólo cambios significativos, que modifican el denominado *gist* de la escena, sean detectados. Por el contrario, los cambios en la fóvea o su zona cercana son detectados rápidamente. Como se puede observar, existe una relación clara entre captura a distintas resoluciones de la escena y atención visual, formando ambas parte de una misma arquitectura de percepción visual, sumamente dinámica y, como se ha comentado ya, eficiente. En su implementación artificial, estas arquitecturas serán las que se clasificaron

en el Capítulo 2 como arquitecturas de visión activa variante en el espacio (*space-variant active vision*) (Schwartz et al., 1995). Entendemos también que un mecanismo de atención que trabaje, no sobre representaciones de resolución uniforme, sino sobre representaciones en las que la cercanía a la fóvea te confiere mayor importancia, se asemejará más a la situación postulada en este mismo párrafo. Como se esbozó en el Apartado 2.3.3 y se analizará en la primera parte de este Capítulo (Sección 3.1), la pirámide irregular foveal cumple este requisito.

Ya se ha repasado en el Capítulo 2 como la captura variante en el espacio del sistema de visión humano ha sido imitada artificialmente por distintas aproximaciones. De igual forma, la atención visual ha sido también estudiada en profundidad, provocando la propuesta de numerosos mecanismos artificiales (Frintrop et al., 2010). Sin embargo, la combinación de atención visual y captura foveal ha sido tratada sólo en limitadas ocasiones, en las que los autores han debido resolver cómo codificar la relación bidireccional entre la estimación de la saliencia visual y la segmentación de la imagen. Una relación normalmente basada en el concepto de objeto perceptivo o proto-objecto (Rensink, 2000), definido vagamente como la unidad de información visual que puede acotarse en una región coherente y estable, y que puede ser extraído de la imagen usando un algoritmo de segmentación perceptivo. El ciclo de trabajo de una arquitectura como las descritas implica, entonces, la segmentación de la imagen en regiones o proto-objetos, cuya saliencia o importancia será evaluada en el marco de un proceso de atención para así determinar a qué nuevo proto-objeto se dirigirá la fóvea. Parecería que es la estimación de saliencia la que depende de la segmentación, pero la relación entre saliencia y segmentación es, como se ha comentado, bidireccional. Para explicar esto volvamos al inicio del ciclo descrito, en el que se segmenta la imagen de entrada, y ahora lancemos el algoritmo de segmentación ¿qué parámetros seleccionar? La segmentación se define en la literatura como un problema abierto (*ill-defined*), pues se obtienen distintos resultados según los parámetros que se apliquen y, en muchas ocasiones, no existe forma de justificar cuáles son los correctos. El problema se muestra en la Figura 3.1, tomada del artículo de Ajay K. Mishra (2012), ¿qué segmentación de las dos mostradas es la correcta? Si estamos interesados en segmentar el caballo que aparece en la escena la solución correcta es la Figura 3.1 (derecha), pero si el objetivo son los árboles en el centro de la imagen, entonces diríamos que la correcta es la Figura 3.1 (centro). La evaluación de la importancia o saliencia es el resultado del mecanismo de atención visual. Así, en estas arquitecturas, una vez se ha estimado la saliencia de toda la imagen y se ha seleccionado una región de interés, la fóvea se mueve para encuadrarla. Esto obliga a capturar una imagen

foveal distinta, cuya segmentación devolverá un nuevo y distinto conjunto de proto-objetos. De esta forma, la última región de interés determina los parámetros de la segmentación, haciendo de ésta un proceso ahora sí correctamente definido (Mishra et al., 2012).

Figura 3-1: Segmentación de una imagen natural en (centro) diez regiones, o (derecha) en sesenta regiones usando el algoritmo del corte normalizado (Imagen de Mishra et al. (2012))

El esquema descrito se muestra en la Figura 3.2. El módulo sombreado se describirá en el Capítulo 4 de esta Tesis, en el que se recogen aspectos más relacionados con la implementación final de todo el sistema en el AP SoC. El resto de módulos se describen en este Capítulo. La arquitectura en sí no está completa: el proto-objeto en la fóvea deberá ser analizado por aquellos componentes de más alto nivel que, condicionados por la tarea en curso, extraigan de él información semántica. Por otro lado, el sistema requiere de un módulo o sistema de inhibición de retorno (Marfil et al, 2014), que asegure que la fóvea no se localice continuamente sobre la misma región de interés.

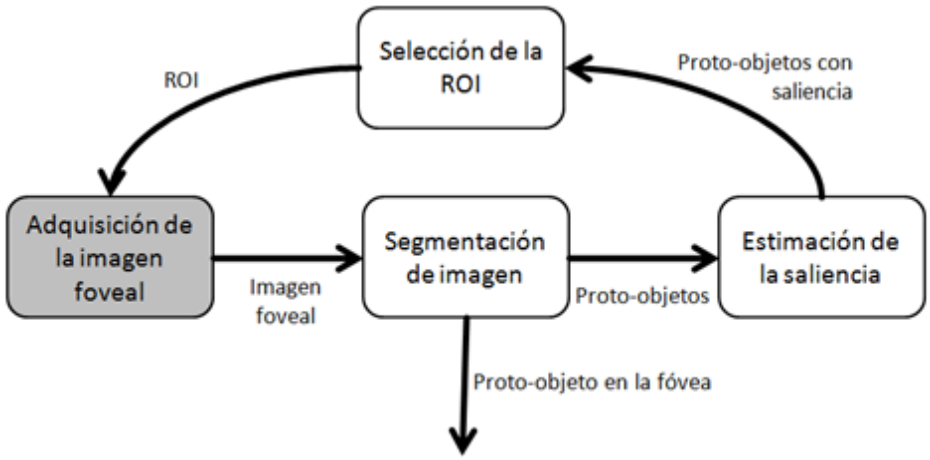


Figura 3-2: Esquema del modelo de atención foveal propuesto

El resto del Capítulo describe, en las Secciones 3.1 y 3.2 los procesos implementados para segmentar la imagen y para estimar la saliencia de cada región extraída. La Sección 3.3 presenta los resultados obtenidos al aplicar el esquema de atención visual completo en algunas secuencias de vídeo. Finalmente, en la Sección 3.4 se discuten los resultados obtenidos.

3.1. Segmentación multirresolución

Como se comentó en el Apartado 2.3.3, la pirámide irregular foveal combina segmentación multirresolución y captura variante en el espacio. Para ello, sobre la imagen foveal se van acumulando niveles que combinan el procesado del nivel inferior, construyendo niveles de mayor abstracción. La construcción de un nivel a partir del inferior es, en sí mismo, un proceso de diezmado, esto es, un proceso de selección de nodos supervivientes, al que se une un proceso de agrupación. Ya se describió la diferencia entre diezmado regular o irregular en el Apartado 2.3.1. En esta Tesis se propone la implementación de un algoritmo de diezmado irregular, basado en el esquema de diezmado dirigido por los datos (D3P) propuesto por Jean Michel Jolion en 2003. Dicha implementación se sintetizará en las partes software y lógica de un dispositivo en chip (SoC) programable (AP SoC), para lo cual será necesario modificarlo como describiremos en el Apartado 3.1.2. Nombraremos esta implementación modificada como *Bounded* D3P, BD3P. La implementación final se describirá en el Capítulo 4.

3.1.1. El algoritmo D3P

Como se describió en el Apartado 2.3.1, el gran problema de las pirámides regulares es que éstas no pueden adaptarse al contenido de la imagen debido a la rigidez de su arquitectura. Como respuesta a este problema surgen, en la década de los 90s, distintos modelos (Meer, 1989; Kropatsch et al., 1993), en los que cada nivel de la jerarquía es ahora un grafo en lugar de una imagen. Aparece el concepto de diezmado que hemos empleado anteriormente para referir cualquier proceso, regular o irregular, de generación de un nivel desde el inferior. Conceptualmente, sin embargo, el diezmado, como selección de un conjunto de nodos supervivientes de entre los nodos de un nivel, es siempre un proceso propio de estructuras irregulares. En las regulares es más frecuente que el nodo padre sea ya parte de la estructura, siendo los enlaces intra-nivel lo único que puede llegar a modificarse para adaptar la estructura al contenido de la imagen.

La primera propuesta de pirámide irregular fue descrita por Peter Meer en 1989. Básicamente la pirámide estocástica ya establece el proceso que definimos en el Apartado 2.3.1 de establecimiento de enlaces y selección de supervivientes. En 1992, dicho esquema será empleado en el caso práctico de la segmentación de imagen por Annick Montanvert y Jean Michel Jolion. Pese a ser la primera propuesta de procesamiento jerárquico adaptada a los datos, su éxito será muy limitado. El mecanismo de diezmado se aplica de forma iterativa sobre el conjunto de datos, en un proceso que debe hacer cumplir la regla de que el resultado del diezmado sea un conjunto independiente maximal del conjunto de nodos del nivel inferior. Esto hace que el proceso de generar cada nivel pueda ser lento y, además, el resultado pueda ser un grafo muy similar en tamaño al del nivel inferior. Para evitar estos problemas, Jean Michel Jolion propone en 2003 el proceso de diezmado dirigido por los datos (*Data Driven Decimation Process*, D3P).

Sea $G^{(l)} = (N^{(l)}, E^{(l)})$ el grafo que representa el nivel l de la jerarquía, donde $N^{(l)}$ define el conjunto de nodos del grafo y $E^{(l)}$ el conjunto de arcos, y sea V_i la vecindad del nodo i definida como $\{j \in V^{(l)} : \delta_{ij}\}$, donde δ_{ij} define el arco $(i,j) \in E^{(l)}$. Definida de esta forma, el nodo i no está incluido en su propia vecindad. Cada nodo i tiene asociado un valor x_i mientras que los arcos no presentan valor alguno. El proceso de diezmado del D3P determina que un nodo i de $G^{(l)}$ sobrevive si y sólo si, es un máximo local o no tiene ningún vecino superviviente. Para implementar este proceso, cada nodo i se caracteriza usando dos nuevas variables binarias, p_i y q_i. La variable p_i marca como 1 aquellos nodos que sobreviven. La variable q_i por su parte, vale 1 en aquellos nodos que, no siendo máximos locales, tampoco tienen un máximo local en su vecindad. Los valores de ambas variables para todos los nodos de la base, $p_i^{(o)}$ y $q_i^{(o)}$, se fijan a 1. Para el resto de niveles se calculan siguiendo las ecuaciones

$$p_i^{(k+1)} \Leftrightarrow (p_i^{(k)} \text{ OR } q_i^{(k)}) \text{ AND } x_i > \max(\{x_j : \delta_{ij} \cdot q_j^{(k)}\})$$

$$q_i^{(k+1)} = [\sim p_i^{(k+1)} \text{ AND } \sim p_i^{(k+1)} \ \forall j \in V_i^{(k)}] \text{ AND } (V_i^{(k)} \neq \emptyset)$$

Lo que implica un proceso ejecutado en dos fases, en la primera se evaluarían los $p_i^{(k+1)}$ y en la segunda los $q_i^{(k+1)}$. Cualquier nodo que, en esta segunda fase, tenga un valor $q_i^{(k+1)}$ igual a 1 también sobrevive, por lo que su valor $p_i^{(k+1)}$ se hace entonces igual a 1. El proceso no es iterativo y puede hacer que muchos nodos, vecinos o no, sobrevivan sin que sean máximos locales.

La Figura 3.3 muestra un ejemplo de aplicación del proceso de diezmado sobre un grafo de 25 nodos, en los que se ha marcado el valor de la variable x_i. En la Figura 3.3 (izquierda) se marcan los nodos supervivientes en rojo (máximos

locales) y en verde (no hay máximos locales en su vecindad). Se observa que sobreviven nodos vecinos, ninguno de los cuales es máximo local.

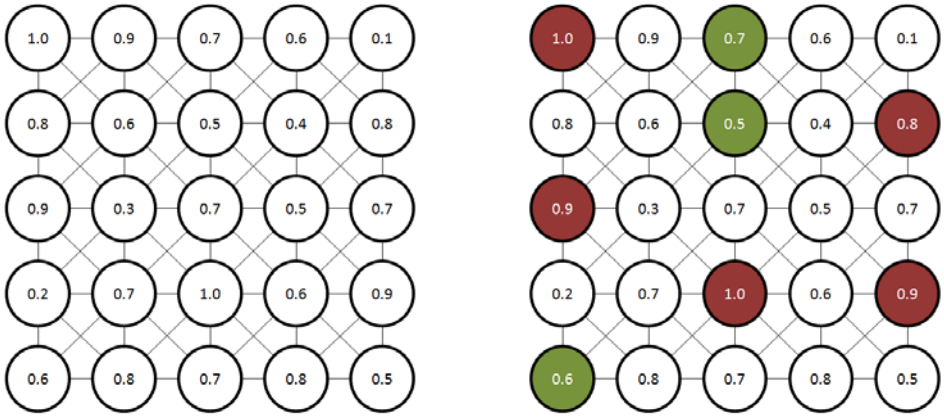


Figura 3-3: (izquierda) Grafo original –en cada nodo se ha marcado el valor de la variable xi-; y (derecha) nodos supervivientes marcados en rojo (máximos locales) y verde.

El proceso D3P se puede aplicar en una única iteración, con las dos fases descritas. En la primera fase se marcan los máximos locales con p_i igual a 1 y q_i igual a 0. En la segunda se evalúan sólo los nodos con q_i igual a 1: aquellos que tienen un máximo local (p_i igual a 1) en su vecindad se marcan con q_i igual a 0. Los nodos con p_i ó q_i igual a 1 se marcan como supervivientes (p_i igual a 1). Si este proceso permite obtener los nodos del nuevo grafo $G^{(k+1)}$, los arcos $E^{(k+1)}$ se estiman como

$$E^{(k+1)} = \{\, (i,j) \in N^{(k+1)} \times N^{(k+1)} : i \neq j \text{ AND } \text{path}(i,j) \leq 3 \,\}$$

donde

$$\text{path}(i,j) = 1 \Leftrightarrow \delta_{ij}^{(k)}$$

$$\text{path}(i,j) = 2 \Leftrightarrow \exists\ y \in N^{(k)} : \delta_{iy}^{(k)} \text{ AND } \delta_{yj}^{(k)} \text{ AND } y \notin N^{(k+1)}$$

$$\text{path}(i,j) = 3 \Leftrightarrow \exists\ y, x \in N^{(k)} : \delta_{iy}^{(k)} \text{ AND } \delta_{yx}^{(k)} \text{ AND } \delta_{xj}^{(k)} \text{AND } y \notin N^{(k+1)}$$

$$\text{AND } x \notin N^{(k+1)}$$

Se aprecia como el cálculo de los arcos del nuevo nivel es un proceso más complejo, algo típico en las estructuras irregulares, que el propio proceso de selección de los supervivientes.

3.1.2. El algoritmo BD3P

El algoritmo D3P presenta dos importantes problemas para su implementación en un sistema empotrado. Por un lado, cada nodo puede tener tantos vecinos como estime la topología de la imagen. Este problema ya fue descrito en el trabajo original de Jean-Michel Jolion en 2003 y es común a las propuestas irregulares. Las soluciones no pasan por limitar estas vecindades sino por ordenarlas usando alguna estrategia, de forma que luego pueda decidirse con quién se debe enlazar cada nodo no superviviente. Por ejemplo, Yll Haxhimusa y Walter Kropatsch propondrán, en las distintas implementaciones de la pirámide de grafo dual, la ordenación de este espacio de enlaces inter-nivel como un árbol, que puede luego cortarse enlazando uno o dos nodos no-supervivientes con cada superviviente. El problema de la asignación dinámica de un número de vecinos a cada nodo en el grafo, relacionado con el trazado del mallado inter-nivel, se convierte en crítico si el grafo que representa cada nivel quiere implementarse en la parte lógica del AP SoC, pero es igualmente importante si se implementa en la parte software y se quieren acotar los tiempos de cómputo y la memoria de almacenamiento. El otro problema grave del D3P radica en la necesidad de calcular, para cada nodo, las distancias L2 y L3 (los $path(i,j)$ iguales a 2 ó 3 que se describen al final del Apartado 3.1.1). Estas distancias complementan la vecindad (distancia L1) y obligarían a almacenar, para cada nodo, quienes son sus vecinos a esas distancias. Su cómputo es iterativo y costoso y su almacenamiento elevaría en exceso el conjunto de información asociado a cada nodo del grafo.

El algoritmo BD3P (*Bounded* D3P) que se propone en esta Tesis aborda ambos problemas y proporciona una implementación más eficiente y rápida del esquema de diezmado D3P, acotando la vecindad de cada nodo usando el marco de aplicación final y eliminando la necesidad de almacenar los vecinos a distancias L2 y L3.

3.1.2.1. Acotación del número de vecinos

En su implementación en la parte lógica programable de un AP SoC, cada nodo del grafo necesitará almacenar el valor de todos sus vecinos. Al desconocerse inicialmente el número de vecinos, éste deberá ser asignado dinámicamente. Esta asignación dinámica, realizada en tiempo de ejecución, básica en el manejo de la memoria en programación, resulta un problema serio en nuestro caso. El problema de la asignación dinámica de recursos en hardware ha sido resuelto mediante la creación de bloques hardware de *Asignación o Desasignación* de memoria (Karabiber et al., 2007). En general, la asignación de bloques de tamaño fijo, que funciona bien en sistemas empotrados muy

simples, no es útil en estos casos, por lo que se suele recurrir al uso de algoritmos *buddy*. Básicamente, un asignador *buddy* gestiona la memoria desde un gran bloque, que suele ser tamaño potencia de dos. Si lo necesario es menor del doble que el tamaño del bloque, éste se parte en dos y el proceso de comparación se repite. El proceso se itera hasta que el bloque se ajuste al necesitado. Esto genera una estructura en árbol, en la que se encajan los bloques de distinto tamaño. Cada rama del árbol se marca con un bit, de forma que al asignar un bloque y poner el bit correspondiente a valor 1, las ramas que cuelguen desde este bloque ya no pueden asignarse. La desasignación consiste en marcar con un valor 0 el bloque ocupado. Esta estrategia ha sido empleada en la implementación de unidades hardware empotradas en diseños SoC. Así, el algoritmo de manejo activo de memoria (*Active Memory Management algorithm*, AMMU) se basa en una modificación del algoritmo *buddy* (Agun y Chang, 2001), en la que se propone el manejo con puertas OR de este árbol de asignación. Es un método fácil de integrar, pero que requiere unas cantidades elevadas de memoria para almacenar los mapas de bits, proporcionales al número de objetos o su tamaño. El algoritmo FEMA, propuesto por Karabiber et al. (2007), se presenta como una alternativa más eficiente, pero igualmente resulta costosa. La Figura 3.4 muestra el diagrama de bloques del algoritmo FEMA. El esquema necesita la implementación de la estructura lógica del árbol de puertas OR, el bloque de búsqueda de espacio libre y el de detección de espacio libre.

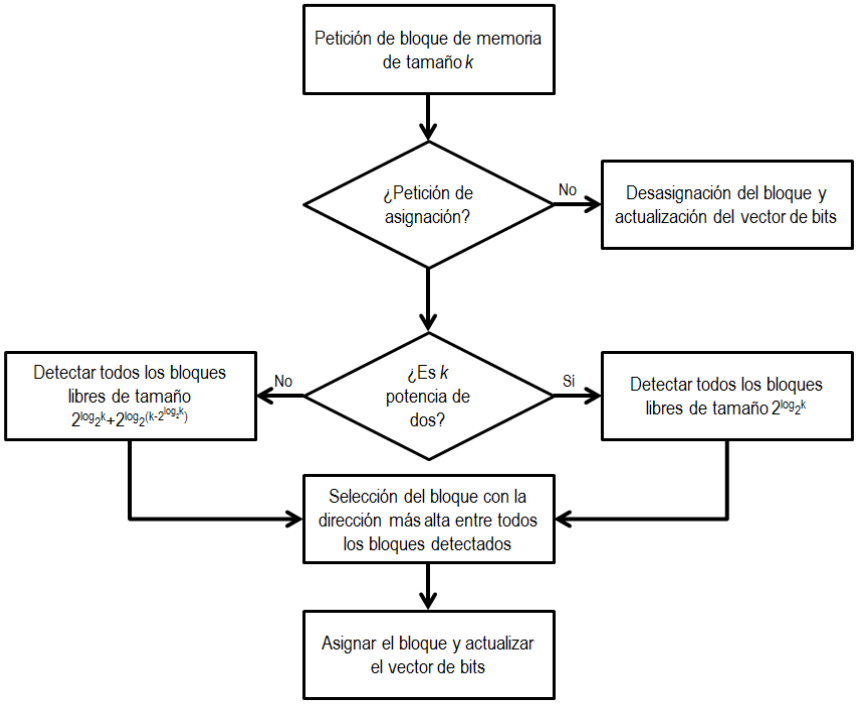


Figura 3-4: Esquema del algoritmo FEMA de asignación dinámica de memoria

Para evitar el consumo de todos estos recursos, se puede optar por asignar un tamaño fijo al vector de vecinos. En la gráfica de la Figura 3.5 se muestra el número de vecinos por nodo y nivel al aplicar el algoritmo D3P a imágenes naturales de tamaño 128 x 128 píxeles, tomadas de la base de datos BSDS500 (Martin et al, 2004). No se aplica ningún factor umbral, por lo que el proceso de agrupamiento termina por agrupar todos los píxeles de la imagen en una única clase. Los datos mostrados están adquiridos del procesamiento de más de 15Mpíxeles. En la gráfica se muestra una franja roja por nivel, obtenida usando el promedio del número de vecinos por nodo y la varianza total de la distribución. Se aprecia como la altura máxima de las pirámides generadas es de seis niveles. También como, en la base, prácticamente todos los nodos tienen ocho vecinos. También que, a partir del nivel 1, el número de vecinos por nodo disminuye. En el nivel 6 prácticamente quedan dos o tres regiones. Observando los resultados parece difícil determinar un valor exacto para el número de vecinos, incluso aunque éste se fije de forma independiente para cada nivel. Es importante además notar que, en una estructura irregular como ésta, el número de niveles es inicialmente desconocido. En función de la topología de la propia imagen, la estructura, con una base de igual tamaño, puede crecer más o menos. Sin embargo, podemos seguir preguntándonos si podemos fijar unos valores máximos. Para ello habrá que analizar más en profundidad el contexto completo, incluyendo la aplicación.

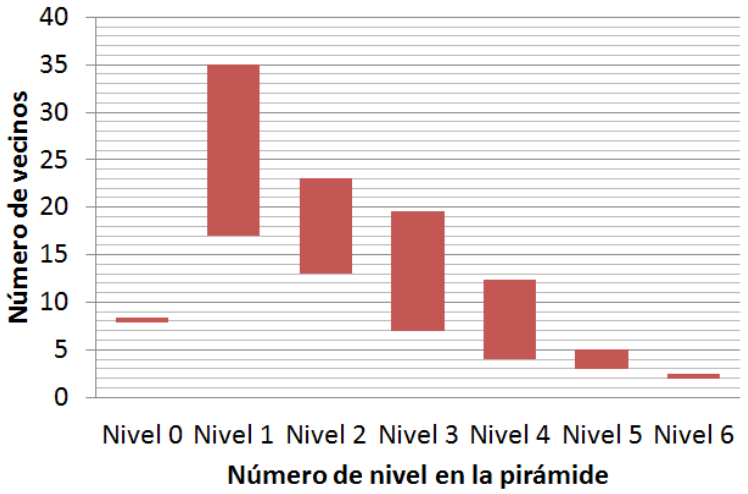


Figura 3-5: Número de vecinos por nodo y nivel en pruebas de segmentación de 1000 imágenes de 128 x 128 píxeles. La franja roja muestra la variación en torno a la media, obtenida usando el promedio y la varianza en la distribución (estas imágenes son recortes de imágenes de la base de datos BSDS500)

3.1.2.2. Proceso de diezmado dirigido por la tarea

En su trabajo original de 2003, J.M. Jolion no tiene en cuenta la aplicación final del proceso de diezmado sino su desarrollo teórico. Si nos movemos al campo de una aplicación concreta, en nuestro caso la segmentación de una imagen color, se puede entender que, en el conjunto de vecinos de cada nodo x, existirá un subconjunto de nodos que nunca se agruparán en la misma ventana de reducción que x al tener un color muy distinto. O si lo expresamos al revés, un subconjunto de nodos que posiblemente terminen por estar en la misma ventana de reducción que x al tener un color muy similar. Ambas afirmaciones, completamente subjetivas, implicarían que es posible limitar el conjunto de vecinos del nodo x a un valor específico. Incluso que dicha limitación puede ser beneficiosa pues, una vez que una conexión, un arco, no se establece entre dos nodos, se está disminuyendo la posibilidad de que ambos formen parte de la misma ventana de reducción. Si por color su fusión no es interesante, al dejar de introducir estos arcos en la estructura se estará potenciando la segmentación de la imagen en un conjunto de clases más limpio. El caso se esquematiza en la Figura 3.6. En la figura de la izquierda, el D3P asocia como vecinos, a todos los nodos del nivel inferior, aquellos que lo son por estar en contacto. Al generar el nivel superior, los dos nodos (rojo y azul) de valor x igual a 0.9 sobreviven, y el nodo azul de valor 0.6 sobrevive al no tener ningún vecino que sobreviva. Al trazar los enlaces inter-nivel, el nodo azul de valor 0.6 situado más a la izquierda sólo es vecino del nodo superviviente rojo, y deberá por tanto enlazarse a éste, degradando el valor de la región asociada a ese padre. El resto de enlaces inter-nivel son correctos. En la figura de la derecha, sin embargo, en el nivel inferior sólo algunos enlaces intra-nivel son válidos, aquellos que se han marcado en trazo más grueso. Ahora, al seleccionar los supervivientes, el nodo azul de valor 0.8 también sobrevive al ser un máximo local (no está conectado al nodo rojo de valor 0.9). Los nodos no supervivientes encuentran ahora vecinos de color correcto en su vecindad y los enlaces son todos correctos. Al plantear acotar el número de vecinos, el método se ha denominado *Bounded* D3P, BD3P. Es importante notar que, junto a la restricción impuesta de similitud en color, también impondremos una restricción de número máximo de vecinos. Esto facilitará la posible implementación en la parte lógica del SoC, pero también mejorará la velocidad de procesado si se ejecuta en la parte software. La primera restricción de distancia en color ayudará a no alcanzar este límite máximo y mejorará el resultado de la segmentación. Las ventajas que sólo se esbozan cualitativamente en la Figura 3.6 se aprecian más claramente en los resultados de segmentación que se presentan en la Figura 3.7. En concreto, en esta Figura 3.7 se ha limitado el número de vecinos a 20. La evaluación de resultados se analiza en más detalle

en el Capítulo 5. En dicho Capítulo se analizará con detalle la aplicación de este algoritmo de segmentación en el marco de la pirámide irregular foveal. La siguiente Sección describe la implementación final del algoritmo del BD3P.

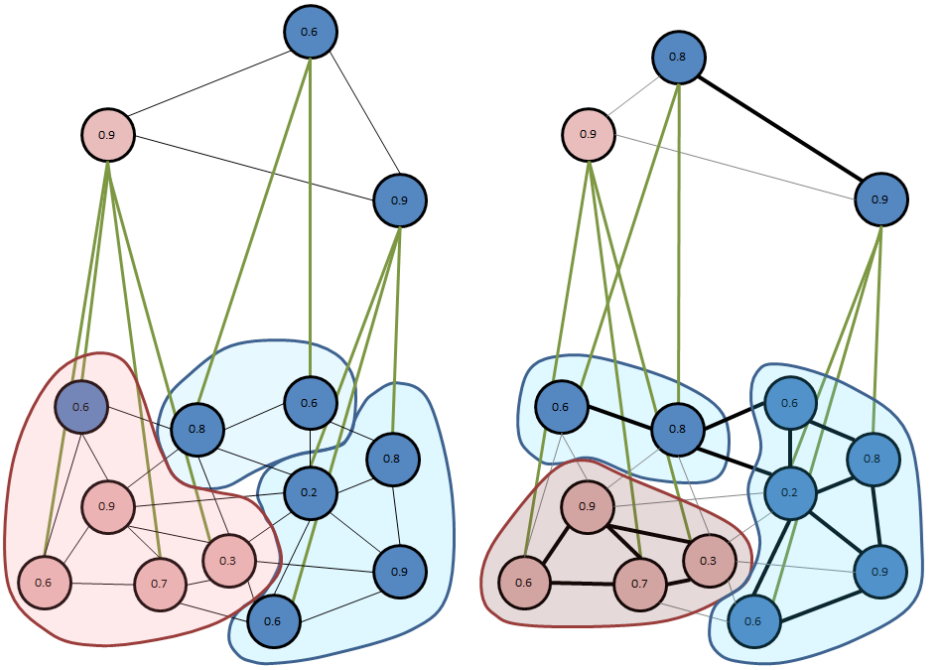


Figura 3-6: Influencia del acotado de enlaces en la estructura interna del D3P (izquierda) D3P original; y (derecha) BD3P.

3.1.2.3. Esquema del algoritmo BD3P

Por todo lo descrito en los apartados anteriores, el esquema del algoritmo BD3P presenta importantes diferencias con respecto al algoritmo D3P original. En gran parte, estas diferencias vienen marcadas por la propia aplicación, como ya ocurriera anteriormente, por ejemplo, con la pirámide adaptativa, implementación práctica en segmentación de la pirámide estocástica de Peter Meer.

A continuación se describe el algoritmo BD3P. Su inclusión en el AP SoC se abordará ya en el Capítulo 4. Los pasos básicos son tres y los pilares la representación por nivel del contenido de la imagen como un grafo, el empleo de la varianza interna del campo receptivo asociado a cada nodo como la característica que determinará su supervivencia, y la ya referida acotación del número de vecinos, tanto por una cota máxima como por la distancia en color entre nodos.

Figura 3-7: Resultados de segmentación: (arriba) imágenes originales; (centro) segmentación usando el D3P; y (abajo) segmentación usando el BD3P

1. Generar el grafo $G[0]$ asociado a la base multirresolución (n=0). La imagen multirresolución se codifica en un vector unidimensional que representa el grafo. El esquema de codificación se ilustra en la Figura 3.8. Básicamente emplea el rexel de mayor tamaño como unidad de desplazamiento vertical y horizontal. Cada una de estas unidades se va descomponiendo en rexeles conforme se va recorriendo la imagen bidimensional. La caracterización en color de cada nodo usará el espacio HSV. En paralelo al proceso de generación de este vector de nodos del grafo, se calculan los vecinos de cada nodo. Se emplea una codificación en vecindad-4.

2. Generar el grafo $G[n+1]$

   a) Selección de los supervivientes de $G[n]$. Cálculo de p y q siguiendo la propuesta original del D3P. Como característica discriminatoria se usará la varianza en color var_x de la ventana de reducción asociada a cada nodo x

$$var_x = \frac{1}{|\lambda_x|^3} \sum_{y \in \lambda_x} (H_y - \bar{H}_x)^2 \sum_{y \in \lambda_x} (S_y - \bar{S}_x)^2 \sum_{y \in \lambda_x} (V_y - \bar{V}_x)^2 \quad (3.1)$$

   donde λ_x representa al conjunto de nodos y que forman la ventana de reducción de x. El valor medio en los tres campos H, S y V, será el valor promediado de color asociado a x. Usando esta métrica, el valor p igual a 1 identificará a aquellos nodos que sean mínimos locales.

   b) Creación de los nodos padres en $G[n+1]$, elevando los nodos supervivientes

   c) Establecer los arcos intra-nivel, enlazando cada nodo no-superviviente con el padre del nivel superior más cercano. Como distancia en color se usará la métrica

$$d(x,y) = |V_x - V_y|^2 + (S_x^2 + S_y^2 - 2 \cdot S_x \cdot S_y \cdot \cos(|H_x - H_y|)) \quad (3.2)$$

   que balancea las componentes acromáticas y cromáticas del color. Se crean las ventanas de reducción λ_x de cada nodo padre en el nivel $n+1$, que incluirán al propio nodo superviviente en el nivel n

   d) Cálculo de los nuevos valores de color y varianza de los nodos padres, empleando las ventanas de reducción y la anterior ecuación (3.1)

   e) Establecer los arcos inter-nivel en el nivel $n+1$. Se evita el cálculo de las distancias L2 y L3 estimando las vecindades en $n+1$ con los datos del nivel n. Dos nodos padre serán vecinos si sus ventanas de reducción están conectadas en el nivel n, la distancia entre esos nodos padre es menor a un umbral y el número de vecinos de cada padre no supera un valor máximo.

3. Repetir 2-3 haciendo $n=n+1$ mientras $G[n+1] \neq G[n]$, $|G[n+1]|>1$ y $n+1<n_{max}$

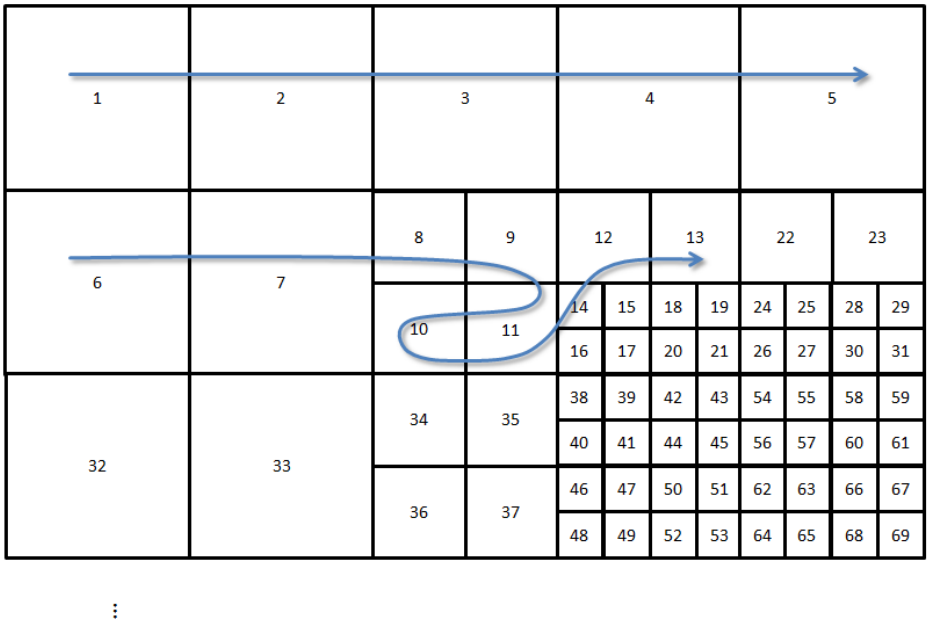


Figura 3-8: Generación del grafo G[0] a partir del layout bidimiensional de sensado multirresolución. Cada número marca la posición del rexel en el vector unidimensional que contendrá el grafo.

3.1.2.4. Selección de parámetros

El BD3P depende de dos parámetros:

- el umbral U_d que determina que dos nodos, que están en contacto en la base de la jerarquía, serán vecinos; y
- el número máximo de vecinos v_{max} por nodo;

que se aplican por las razones ya expuestas en los Apartados 3.1.2.1 y 3.1.2.2. Ambos parámetros limitan el establecimiento de los enlaces inter-nivel, pero cabe ahora preguntarse cómo influye cuantitatívamente esta acotación en los resultados de segmentación y también cómo de fácil es establecer unos valores que resulten adecuados.

Para valorar los resultados obtenidos con el BD3P en segmentación se usará la función Q (Borsotti et al., 1998). Esta función tiene en cuenta, como indicadores de bondad, (i) que las regiones sean uniformes y homogéneas en color, (ii) que las regiones no presenten mucho hueco interno, (iii) que regiones adyacentes

presenten valores significativamente distintos en color, y (iv) que no existan excesivas regiones pequeñas.

$$Q(I)=\frac{1}{1000\,(N\cdot M)\sqrt{R}\sum_{i=1}^{R}\left[\frac{e_i^2}{1+\log A_i}+\left(\frac{R(A_i)}{A_i}\right)^2\right]}$$

siendo N x M el tamaño de la imagen y R el número de regiones en la segmentación. A_i y e_i son el área y error en color medio de la región i, respectivamente. $R(A_i)$ es el número de regiones en la segmentación que tienen el mismo área que A_i.

La Tabla 3.1 presenta la comparación de la BD3P con otras estrategias de diezmado. Así, en la Tabla se pueden encontrar métodos regulares, como la pirámide enlazada (LRP) (Burt et al., 1981) o la probabilística (WRP) (Hong y Rosenfeld, 1984), e irregulares, como el D3P (Lallich et al, 2003), la pirámide irregular acotada (BIP) (Marfil et al, 2006), la jerarquía de particiones (HIP) (Haxhimusa y Kropatsch, 2004), y la combinatoria (CIP) (Brun y Kropatsch, 2003). La evaluación cuantitativa se ha llevado a cabo usando 50 imágenes color de la base de datos Coil-100[7] (alguna de ellas se muestran en la Figura 3.7). Todas las imágenes se han redimensionado a tamaño 256 x 256 píxeles. Los métodos se han ajustado para obtener los mejores resultados en el indicador Q.

	Q			Altura de la jerarquía			Número de regiones		
	Q_{min}	Q_{med}	Q_{max}	h_{min}	h_{med}	h_{max}	N_{min}	N_{med}	N_{max}
LRP	1052.1	1570.3	2210.3	9	9	**9**	17	81.4	208
WRP	1133.7	1503.5	2080.8	9	9	**9**	18	79.6	**149**
D3P	355.6	**818.5**	1301.1	11	32.9	64	45	107.6	201
BIP	**343.2**	1090.9	1911.3	**8**	**8.7**	15	**8**	83.6	230
HIP	460.5	955.1	1530.7	9	11.4	19	23	**76.1**	150
CIP	430.7	870.2	**1283.7**	9	74.2	202	24	91.2	238
BD3P	412.6	831.5	1497.1	9	13	18	37	97.6	178

Tabla 3-1: Valor de la función Q, altura de la jerarquía y número de regiones para distintos esquemas de diezmado (ver texto)

La Tabla 3.1 muestra como los métodos irregulares obtienen mejores valores del índice Q que los regulares. Dentro de los irregulares, el D3P y el BD3P presentan valores similares, algo inferiores a los del CIP.

7 http://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php

Respecto a la sensibilidad en la elección de los parámetros, tanto el umbral U_d como v_{max} pueden fijarse a valores muy altos, y entonces el comportamiento del BD3P se acercará al D3P. En este caso, se han fijado valores que, aún aumentando ligeramente el valor de Q respecto al D3P, han permitido reducir la altura de la jerarquía, cuestión asociada a la mejora de la eficiencia. Las ventajas adicionales para la integración en el AP SoC que proporcionan el acotamiento del número de vecinos compensan la pequeña diferencia en el índice Q.

Se volverá sobre este tema del ajuste de parámetros en el Apartado 5.2.1, pero ya en un marco de aplicación muy concreto: la segmentación de imágenes naturales de la base de datos BSDB500.

3.2. Estimación de la saliencia

Una vez que la escena se ha dividido en regiones o proto-objetos, el siguiente paso es la selección del más relevante. De acuerdo a los trabajos de Anne Treisman y Garry Gelade (1980), este proceso se basa en el cálculo de un conjunto de características de bajo nivel. Pero ¿qué características deben tomarse en consideración?, ¿qué características guían realmente la atención?

Los psicólogos nos dicen que características como el color (Treisman y Souther, 1985), el movimiento (McLeod et al., 1988) o la orientación (Wolfe et al., 1992) influyen claramente en la estimación de la saliencia. De hecho, Jeremy Wolfe y Todd Horowitz (2004) afirman que estas tres características, junto al tamaño, son las únicas que, sin lugar a dudas, deberían evaluarse en el marco de un mecanismo de atención. En este trabajo, Jeremy Wolfe ofrece una lista de características que podrían emplearse para guiar la atención, clasificadas en función de su grado de importancia. El conjunto es bastante amplio y, aunque es evidente que calcular un conjunto enorme de propiedades de bajo nivel puede aportar riqueza a la descripción, el tiempo de cómputo puede ser inaceptable. Claramente es necesario establecer un equilibrio entre eficiencia computacional y el número y tipo de las características seleccionadas.

Siguiendo estas premisas, en este sistema se emplearán cinco características para estimar la saliencia. Del conjunto de atributos que Jeremy Wolfe cataloga como indudables se han seleccionado el color, el tamaño y la orientación. Dado que nuestro algoritmo de segmentación no distingue fondo y objetos, las regiones de mayor área se asocian, normalmente, a regiones sin relevancia tales como paredes o suelos. Utilizaremos el tamaño como un indicador de que la región pertenece al fondo. También se empleará para despreciar regiones de

tamaño muy reducido, que aparecerán normalmente en la fóvea o cerca de ella. No se ha implementado un módulo de detección de movimiento o flujo óptico de bajo nivel. Es por ello que el movimiento ha sido inicialmente descartado. Aunque el contraste en intensidad no es considerado un atributo indispensable, esta característica ha sido incluida en nuestro esquema como un caso especial de contraste color (la intensidad aborda el caso de los valores grises, incluido el blanco o el negro). Además, hemos añadido una característica más relacionada con la forma del proto-objeto: la redondez, parámetro típicamente propuesto como representativo de auténticos objetos. La saliencia final de un proto-objeto, sal_i, vendrá finalmente determinado por la suma ponderada de estas características

$$sal_i = \vec{\lambda} \cdot \vec{f}$$

donde λ es el conjunto de pesos, normalizados para que su suma sea la unidad, y f es el vector de características, que calcularemos como se describe en las siguientes subsecciones. Como se propone en la reciente Tesis Doctoral de Antonio J. Palomino (2014), estos pesos pueden adaptarse a la tarea en curso, aumentando la saliencia de los proto-objetos o regiones de la escena que puedan ser de mayor interés para resolverla.

3.2.1. Influencia en los cálculos del procesado multirresolución

En los siguientes apartados se describirá cómo se calculan las características que determinan la atención. En las fórmulas que determinan estas características se usan perímetros, medias o momentos. Antes de presentar esas fórmulas hay que precisar cómo influye en su cómputo el hecho de tratar con una imagen multirresolución.

El problema se describe en la Figura 3.9, en la que se ha marcado en azul una posible región, repartida entre distintos anillos de resolución. El perímetro de la región, o la longitud del mismo que ésta comparte con las cuatro regiones que la rodean, no puede calcularse sumando cuántos píxeles hay en el borde de la región, pues ahora la región se caracteriza por réxeles de distintos tamaños. En el ejemplo de la figura, la región azul presenta un perímetro b_i

$$b_i = 5 \cdot L + 7 \cdot L/2 + 4 \cdot L/4$$

que dependerá del distinto tamaño de los réxeles (en este caso, L es el tamaño del mayor de los réxeles que forman parte de la región), cuestión que habrá que tener en cuenta al establecer quiénes son los vecinos de la región. Además, como el algoritmo BD3P no almacena toda la vecindad del nodo, sino

sólo aquellos que son similares en color, habrá que mantener una doble estructura de almacenamiento en memoria para guardar esta información.

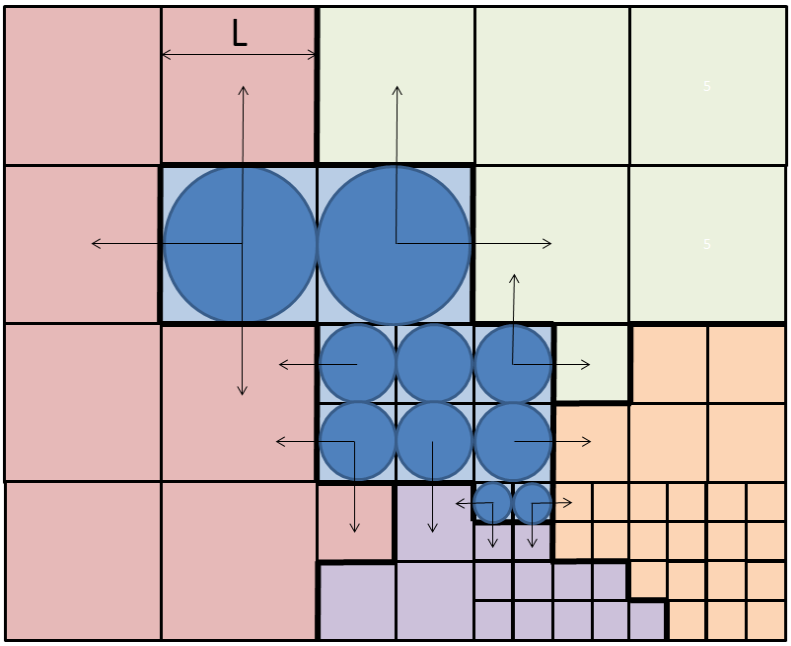


Figura 3-9: La región en la imagen multirresolución podrá estar formada por réxeles de distintos tamaños. Eso influirá en los cálculos de perímetros, momentos o medias (ver texto).

En el caso del cálculo del centroide o centro de masas de la región, habrá también que tener en cuenta el distinto peso de cada uno de los réxeles que componen la región. Esto es

$$(\bar{x}, \bar{y}) = \frac{1}{\sum_i w_i} \sum_i w_i (x_i, y_i)$$

donde w_i es el valor que determina el peso de cada rexel (x_i, y_i) y vendrá dado por el área del mismo. El valor (x_i, y_i) deberá ser el centro de masas del rexel.

Finalmente, también el cálculo de cualquier momento deberá venir ponderado por este peso. Por ejemplo, para el momento $\mu_{1,1}$

$$\mu_{1,1} = \sum_i w_i (x_i - \bar{x})(y_i - \bar{y})$$

3.2.2. Contraste color y contraste intensidad

Los contrastes color e intensidad miden como de diferente es un proto-objeto de las regiones que lo rodean en la imagen en lo que se refiere a color e intensidad (luminosidad). Al formar parte de la teoría de atención de Anne Treisman y Garry Gelade, estas características han sido empleadas en los sistemas artificiales de atención desde los primeros modelos computacionales (Itti et al., 1998).

Dado que los proto-objetos son el resultado de un proceso de segmentación perceptivo, el contraste color (*ColCON*) de un proto-objeto específico se puede calcular como la media del gradiente de color evaluado a lo largo de toda la periferia del proto-objeto. Para acelerar el cálculo y reducir la complejidad de su cómputo, dicho gradiente se calcula usando el color medio de las regiones:

$$ColCON_i = \frac{S_i}{b_i} \sum_{j \in N_i} b_{ij} \cdot disColor(C_i, C_j)$$

donde b_i es el perímetro del proto-objeto P_i, N_i es el conjunto de proto-objetos que son vecinos de P_i, b_{ij} es la longitud del perímetro de P_i en contacto con el proto-objeto P_j, $disColor(C_i,C_j)$ es la distancia color HSV entre los valores medios de color de los proto-objetos P_i y P_j; y S_i es el valor medio de saturación del proto-objeto P_i. Para evitar el cálculo intermedio de los contactos entre proto-objectos y los correspondientes b_{ij}, el proceso de cálculo del contraste color se lleva a cabo no atendiendo a regiones, sino a réxeles. Eso permite calcularlo recorriendo sólo los réxeles del proto-objeto

$$ColCON_i = \frac{S_i}{b_i} \sum_{j \in M_i} \sum_{k \in N_j \wedge k \notin M_i} b_{jk} \cdot disColor(C_j, C_k)$$

donde M_i define el conjunto de réxeles que forman al proto-objeto P_i, y N_j al conjunto de vecinos del rexel j. Los valores de b_{jk} serán la longitud del lado del rexel, como se muestra en la Figura 3.9, un valor conocido. El valor de color del rexel, C_j, coincide con el del proto-objeto P_i.

Debido al uso de S_i en la ecuación de contraste color, los proto-objetos de color blanco, negro o gris no son tenidos en cuenta (su contraste color es siempre pequeño). Por ello se ha introducido una medida específica del contraste en intensidad. El contraste en intensidad (*IntCON*), de un proto-objeto, P_i, se calcula como el gradiente de valores medios de intensidad calculados a lo largo de su borde:

$$IntCON_i = \frac{1}{b_i} \sum_{j \in N_i} b_{ij} \cdot disBrillo(I_i, I_j)$$

donde I_i es el valor medio de luminosidad del proto-objeto. Al igual que con el contraste color, esta fórmula se ha modificado en la práctica para trabajar con los réxeles que forman el proto-objeto.

Como resultado de este proceso se obtienen dos mapas de evidencias basados en el concepto de contraste, donde aquellos proto-objetos que difieren más en color, o intensidad, con las regiones que la rodean se marcan como más relevantes.

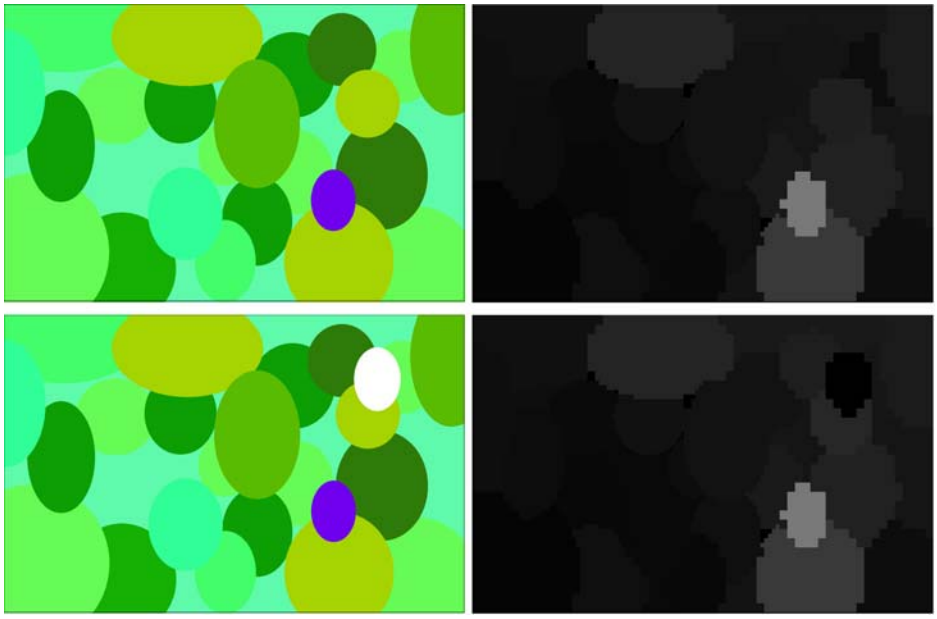


Figura 3-10: Ejemplo de cálculo de las características de contraste color: (izquierda) imágenes originales (1920 x 1024 píxeles); y (derecha) mapa de evidencias asociado al contraste color. La fóvea en ambas imágenes se ubica usando T=B=7, L=8 y R=20.

La Figura 3.10 muestra dos ejemplo de mapas de evidencia de contraste color. En la figura superior, sólo una de las regiones presenta una tonalidad distinta a la dominante, situación que es correctamente detectada por esta característica, como se muestra en el mapa de evidencia. En la figura inferior, sin embargo, hay una segunda región con una tonalidad distinta a la mayoría. Esta región, que presenta un color blanco, no es detectada correctamente por la medida de contraste color.

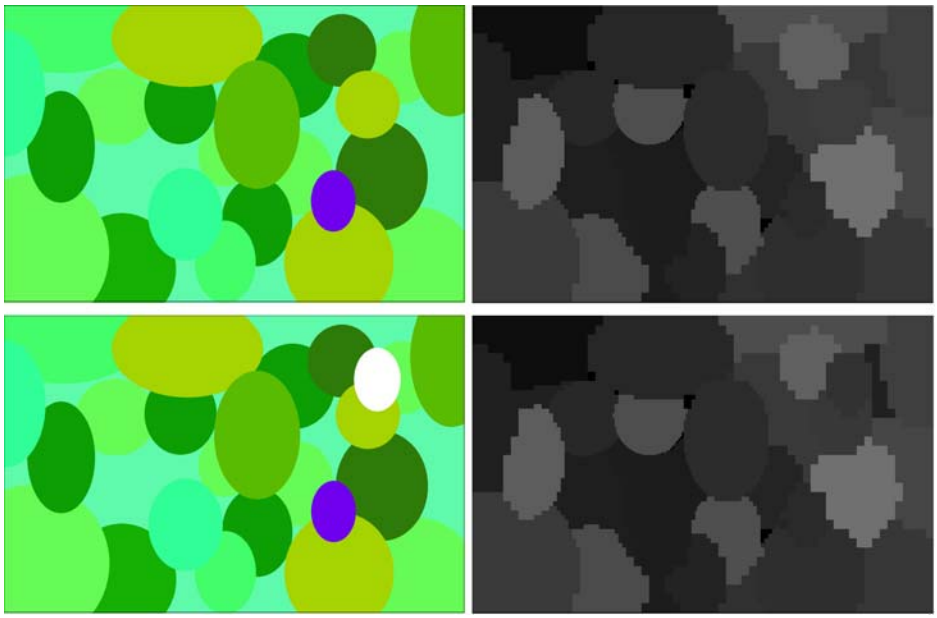


Figura 3-11: Ejemplo de cálculo de las características de contraste intensidad: (izquierda) imágenes originales (1920 x 1024 píxeles); y (derecha) mapas de evidencias asociado al contraste intensidad. La fóvea se ubica en ambas imágenes usando T=B=7, L=8 y R=20.

El problema que ilustra la Figura 3.10 se soluciona con el contraste en intensidad. La Figura 3.11 muestra los mapas de evidencias asociados al contraste en intensidad para las dos imágenes de entrada empleadas como ejemplo en la Figura 3.10. La región blanca no es la más relevante en la imagen inferior pues alguna de las regiones que la rodean presentan también valores claros, pero se puede apreciar como ahora no es la tonalidad la que determina la saliencia sino el brillo (la mayoría de las regiones más salientes son regiones oscuras, en unas imágenes mayoritariamente claras).

3.2.3. Redondez

La medida en redondez refleja lo parecido a un círculo que es el proto-objeto. Esta característica proporciona información sobre la convexidad, cierre y dispersión del proto-objeto. La redondez se obtiene empleando la clásica técnica basada en los momentos. En concreto se usan tres momentos centrales diferentes:

$$\mu^{i}_{1,1} = \sum_i w_i (x_i - \bar{x})(y_i - \bar{y}) \quad \forall (x_i, y_i) \in P_i$$

$$\mu^{i}_{2,0} = \sum_i w_i (x_i - \bar{x})^2 \quad \forall (x_i, y_i) \in P_i$$

$$\mu^i_{0,2} = \sum_i w_i (y_i - \overline{y})^2 \quad \forall (x_i, y_i) \in P_i$$

donde $(\overline{x}, \overline{y})$ es el centro del proto-objeto P_i.

Combinando las ecuaciones anteriores es posible medir la diferencia entre una región determinada y un círculo perfecto. La medida, conocida como excentricidad, se calcula como

$$ecc_i = \frac{(\mu^i_{2,0} - \mu^i_{0,2})^2 + (2 \cdot \mu^i_{1,1})^2}{(\mu^i_{2,0} + \mu^i_{0,2})^2}$$

estando el resultado acotado al intervalo [0 . . . 1].

Finalmente, la redondez ($ROUND_i$) de un proto-objeto se obtiene a partir de la excentricidad usando la ecuación:

$$ROUND_i = 1 - ecc_i$$

La Figura 3.12 muestra un ejemplo del mapa de evidencias asociado a esta característica. Los mapas de saliencia que se muestran en la figura muestran que el proceso de fovealización distorsiona la forma de aquellas regiones que están en la zona periférica pero que, aún así, los círculos y cuadrados en la imagen obtienen los valores de saliencia más altos (mostrados en la figura como tonos más claros). Las formas más alargadas muestran valores menores de saliencia. La figura muestra los mapas de saliencia tras dos sacádicos, que han movido la fóvea, primero, al cuadrado verde, y después al círculo azul. Éste no tiene el valor máximo de saliencia en primera instancia debido a la distorsión que sufre su forma en la primera fovealización.

3.2.4. Orientación

Como se comentó al comienzo de la Sección 3.2, la orientación es una de las fuentes de información que debe incluirse en el cálculo de la atención según Jeremy Wolfe. La orientación de una región en la imagen puede también calcularse usando los momentos centrales anteriormente descritos:

$$\phi_i = \frac{1}{2} \arctan\left(\frac{2\mu^i_{1,1}}{\mu^i_{2,0} - \mu^i_{0,2}}\right)$$

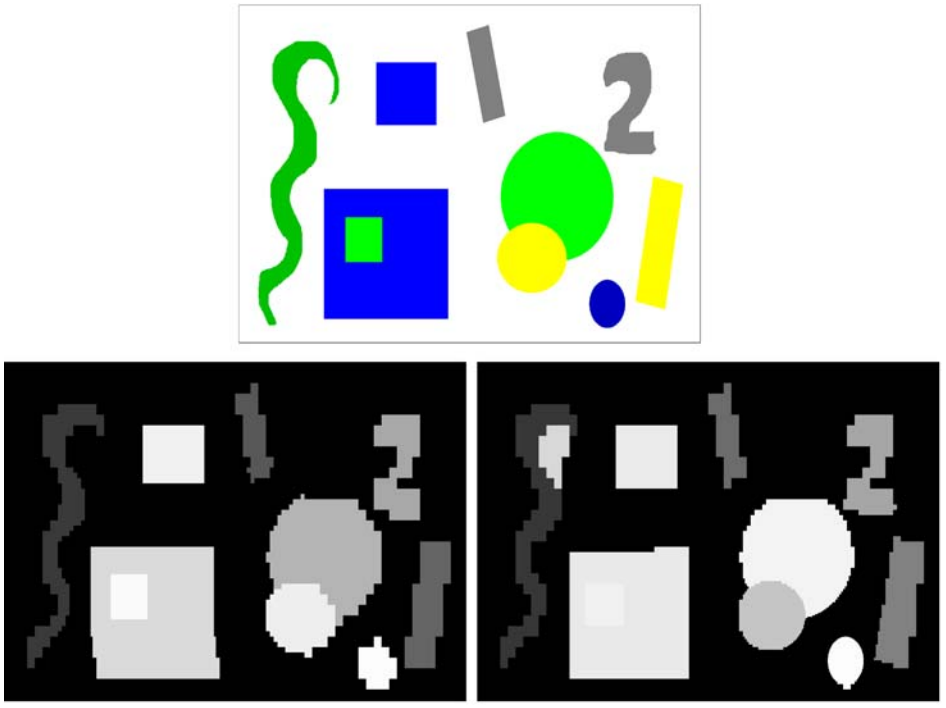


Figura 3-12: Ejemplo de cálculo de la característica de redondez: (arriba) imagen original de 1920 x 1024 píxeles; y (abajo) mapas de evidencias asociados a la redondez de la regiones tras mover la fóvea por dos veces al proto-objeto más relevante. Los proto-objetos más relevantes se muestran en tono más brillante. En la primera fovealización, los parámetros de la GMFD de tamaño adaptativo son T=B=7, L=8 y R=20. En la segunda, la fóvea se desplaza al proto-objeto más relevante: el cuadrado verde dentro del azul. En la tercera al círculo azul. Estas segunda y tercera fovealización generan los mapas de evidencias que se muestran en esta figura.

Pero la orientación de un proto-objeto, en sí misma, no proporciona ninguna información práctica sobre su relevancia. Sólo cuando se compara la orientación de un proto-objeto con la del resto de proto-objetos en la imagen se puede obtener una medida de relevancia. Por todo ello, es más interesante calcular la saliencia en términos de contraste. El contraste en orientación ($OriCON_i$) de un proto-objeto se obtiene usando la ecuación:

$$OriCON_i = \sum_j |\phi_i - \phi_j|$$

donde j es el conjunto de proto-objetos en la imagen.

La Figura 3.13 muestra un ejemplo del cálculo de la característica contraste en orientación. La mayoría de los objetos están orientados en una determinada

dirección, excepto uno de los rectángulos, que presenta una orientación distinta. El primer mapa que se muestra en la figura representa los valores de saliencia cuando la fóvea se ubica en una posición aleatoria, sobre uno de los rectángulos orientados en la dirección mayoritaria. Este mapa ya muestra como el proto-objeto más relevante es el que se orienta en una dirección distinta a la mayoría. El siguiente mapa muestra los valores de saliencia cuando la fóvea se ubica sobre este proto-objeto. La relevancia de esta región, en lo que a contraste en orientación se refiere, aumenta incluso respecto a los valores obtenidos en el mapa anterior.

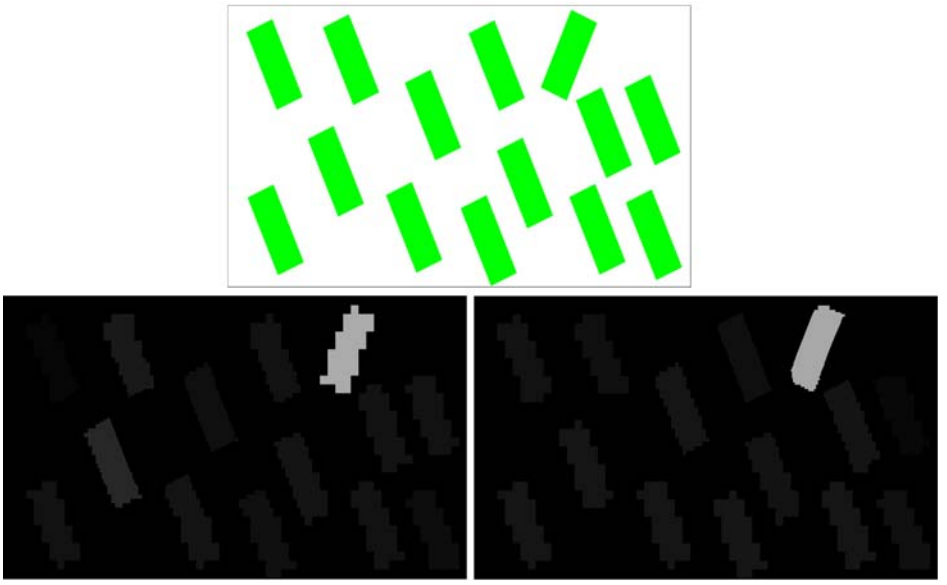


Figura 3-13: Ejemplo de cálculo de la característica de contraste en orientación: (arriba) imagen original de 1920 x 1024 píxeles; y (b) ; y (abajo) mapas de evidencias asociados al contraste en orientación de la regiones tras mover la fóvea desde una posición aleatoria al proto-objeto más relevante. Los proto-objetos más relevantes se muestran en tono más brillante. En la primera fovealización, los parámetros de la GMFD de tamaño adaptativo son T=B=7, L=8 y R=20. En la segunda, la fóvea se desplaza al proto-objeto más relevante: el rectángulo verde con distinta orientación.

3.2.5. Evaluación del mecanismo de atención

El mecanismo de atención ha sido evaluado usando la base de datos Toronto (Bruce y Tsotsos, 2009), definida como la más empleada en el buen artículo de revisión de la materia de Ali Borji y Laurent Itti (2013b). La base de datos contiene 120 imágenes de 681 x 511 píxeles, con información de seguimiento

de ojos de veinte personas. Estas personas vieron la imagen durante cuatro segundos, sin que se le especificara tarea alguna. La Figura 3.14 muestra distintas imágenes de la base de datos. Las fijaciones se han dibujado sobre las imágenes. Basándose en estos puntos de fijación, se ha generado un mapa de densidad de fijaciones para cada imagen (Bruce y Tsotsos, 2009). Se muestran también en la Figura 3.14, bajo cada imagen original.

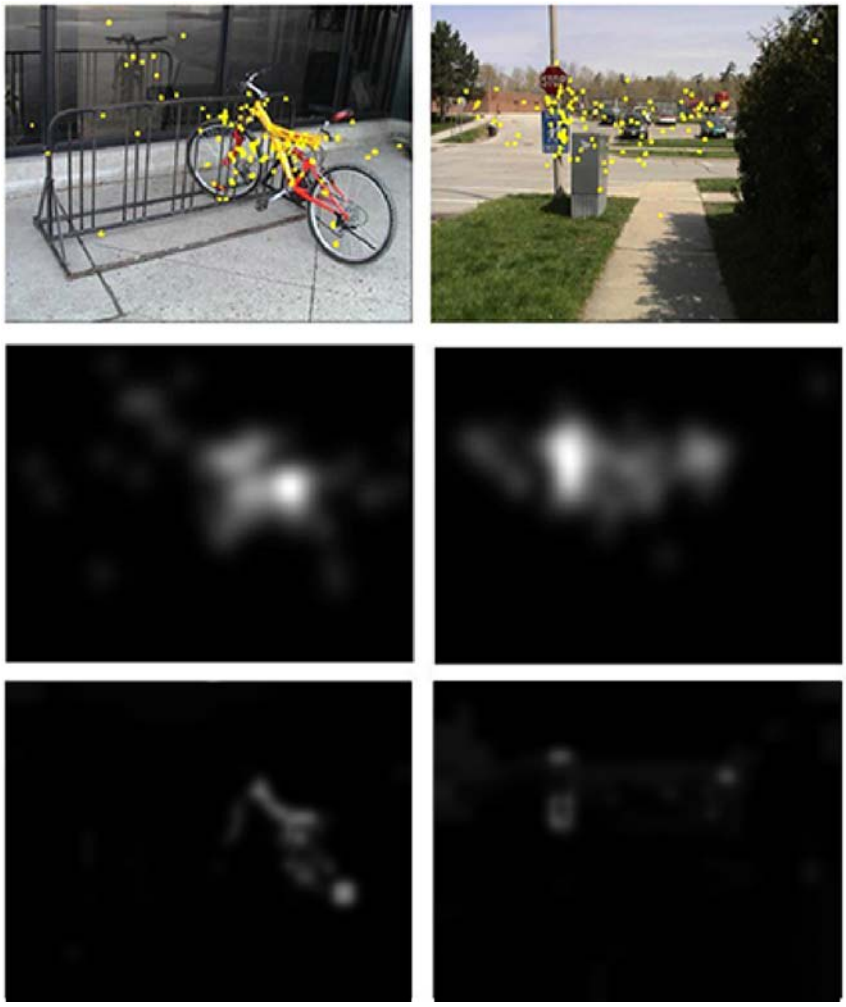


Figura 3-14: (arriba) Imágenes de la base de datos Toronto anotadas con los puntos de fijación (ver texto); (centro) mapas de densidad de fijaciones obtenidos desde los puntos de fijación de veinte personas; y (abajo) mapas de densidad de fijación obtenidos por el esquema de segmentador y mecanismo de atención propuestos.

En la Figura 3.14 se muestran, en la fila inferior, los mapas de densidad de fijación obtenidos usando el sistema propuesto. Para obtener estos mapas se ha fovealizado con un esquema multirresolución de cinco anillos (réxeles de 32, 16, 8, 4 y 2 píxeles de lado) y una fóvea de 176 x 140 píxeles. Las imágenes se han redimensionado a 672 x 512 píxeles, para hacer ambas dimensiones divisibles por 32. El umbral de parecido en color se ha fijado experimentalmente para obtener buenos resultados, pero se ha dejado luego fijo para todas las imágenes de la base de datos. El efecto de suavizado que presentan los mapas de densidad se ha conseguido filtrando, con una Gaussiana, los mapas de saliencia. En este apartado se proporciona una simple evaluación cualitativa del mecanismo de atención, que será completada con datos cuantitativos en el Capítulo 5 de esta Tesis.

Frente a la mayoría de los métodos de atención visual, nuestros mapas de saliencia sí que pueden ser estimados desde un conjunto de fijaciones. Sin embargo, en lugar de obtenerse desde un conjunto de puntos de fijación, como se hace con los seguidores de la mirada aplicados a personas, nuestras fijaciones se asocian a regiones y no a puntos bidimensionales de la imagen. Así, los mapas de densidad de fijaciones que se muestran en la fila inferior de la Figura 3.14 se han construido a partir del suavizado de la suma de las *n* regiones más salientes. El número *n* es igual a la media de las fijaciones humanas almacenadas para esta imagen en la base de datos original. En la Figura 3.15 se muestran datos parciales que permiten determinar como se obtiene uno de estos mapas de densidad.

3.3. Discusión

Los modelos de atención que se presentan en la literatura suelen centrarse en aspectos normalmente relacionados con la identificación de las características que deben influir en la propia atención, con la combinación de estas características para determinar el mapa final de saliencia, o con cómo una tarea específica puede guiar las fijaciones. Pero en general suelen despreocuparse de la naturaleza foveal del sistema de atención humano, en el que dicen normalmente fundamentarse. Los pocos métodos que siguen una estrategia multirresolución suelen emplear para ello dos imágenes, capturadas desde distintas cámaras (Meger et al., 2008): una de baja resolución permite calcular el mapa de saliencia asociado a la escena y otra de alta resolución permite capturar con detalle la región de mayor relevancia. El empleo de una fóvea que sólo captura ciertas regiones de la imagen ha sido propuesto como un método eficiente para comprimir el contenido de la imagen (Geisler y Perry, 1998; Guo y Zhang, 2010). Construido sobre la propuesta de codificación

foveal de Geisler y Perry (1998), el GAFFE (*Gaze Attentive Fixation Finding Framework*) emplea cuatro características locales de bajo nivel de la imagen (iluminación, contraste y un filtrado paso banda de estas dos) para construir el mapa de saliencia y decidir dónde fijar el foco de atención (Rajashekar et al., 2008). Reemplazando estas características por otras empleadas en modelos más recientes, como el AIM (Bruce y Tsotsos, 2009) o el SUM (Zhang et al., 2008), Gide y Karam (2012) han evaluado y mostrado como el empleo de un esquema foveal permite mejorar los resultados ofrecidos inicialmente por estos métodos.

Figura 3-15: (arriba) Imagen #67 de la base de datos Toronto y segmentación proporcionada por el BD3P (fóvea centrada de 176 x 140 píxeles y cinco anillos de resolución); y (abajo) mapa de saliencia proporcionado por el mecanismo de atención propuesto (λ={1,1,1,1} y mapa de densidad de fijación obtenido del procesado del mapa de saliencia anterior con un filtrado Gaussiano (σ=10.0).

En este trabajo la región de interés se extrae tras evaluar la saliencia de todos los proto-objetos devueltos por un proceso de segmentación de la imagen. Estamos por tanto ante un proceso novedoso, pues los citados trabajos de Rajashekar et al. (2008) o Gide y Karam (2012) determinan el punto de la siguiente fijación evaluando características de la propia imagen. Además, en

nuestra propuesta, al mover la fóvea y enmarcar la región más relevante, toda la captura de la imagen cambia, lo cual determina un mapa de saliencia totalmente diferente, que normalmente determinará un nuevo foco de atención en una posición cercana a la ya estudiada. Sólo características muy llamativas en la periferia moverán la fóvea bruscamente hacia ésta, en un comportamiento que se asemeja al que se percibe en el sistema humano.

Finalmente destacar que la propuesta de segmentación descrita en este Capítulo permite obtener resultados de mucha calidad en un esquema que puede ser sintetizado en las partes lógicas y software del AP SoC. Tanto la evaluación de este algoritmo como del propio sistema de atención foveal se analizan en el Capítulo 5.

Capítulo 4

Implementación en el AP SoC

Para muchas aplicaciones, diseñar el sistema completo en una plataforma hardware tipo FPGA, no es la solución más práctica. En el caso concreto de los sistemas de visión artificial, incluso los algoritmos de procesamiento más intensivos contienen frecuentemente secciones secuenciales que pueden ser implementados más fácilmente en procesadores. Las soluciones de co-diseño hardware/software buscan combinar lo mejor de estos dos entornos, haciendo uso de la facilidad de programación de los procesadores pero diseñando módulos hardware para tratar con las partes de mayor consumo de la aplicación.

La inclusión de cores de procesadores empotrados en la lógica programable ha hecho de las FPGAs una plataforma excelente para tratar con estas aplicaciones. Durante el año 2011, los dos mayores fabricantes de FPGAs (Xilinx y Altera) anunciaron el lanzamiento de nuevas familias de dispositivos que combinaban procesadores ARM relativamente potentes con lógica programable de bajo consumo. Aunque ya se habían fabricado dispositivos con procesadores on-board, estas nuevas familias presentaban como novedad que el ARM constituía ahora, más que la propia lógica programable, el centro de todo el dispositivo. Este hecho fortalecía indudablemente el ya creciente avance hacia soluciones de co-diseño centradas en el procesador, y englobaba en esta dirección a los dispositivos basados en la FPGA. En un estudio de 2012 realizado por Embedded Market Survey, más de un tercio de los ingenieros encuestados que aún no usaban FPGAs en su diseños confesaban que, estas nuevas plataformas, podría forzarlos a reconsiderar su empleo.

En esta Tesis se aborda fundamentalmente la implementación de un sistema de procesamiento de imagen complejo, cuya algorítmica se ha presentado en los Capítulos 2 y 3, en una de estas nuevas plataformas, la Zynq®-7000 AP SoC XC7Z020-CLG484-1 de Xilinx. En concreto, se usará como soporte la placa de desarrollo Zedboard de Avnet que, además del AP SoC, dispone de una memoria externa DDR3 de 512 MB. Esta memoria será básica para poder abordar la implementación de un diseño que por sus características, y pese a los esfuerzos ya referidos en el Capítulo 3, consumirá enormes recursos de almacenamiento.

El presente Capítulo se ha dividido en tres apartados, que recogen respectivamente la descripción general del sistema y su particionado en partes lógica y software, la descripción de la implementación en la parte lógica de los módulos hardware, y la descripción de la implementación software de aquellos módulos de funcionamiento mayormente secuencial.

4.1. Descripción general de la implementación

En la introducción del Capítulo 3, en la Figura 3.2, ya se presentaba un diagrama muy general de la implementación global del sistema de visión. Con él se justificaba la integración de un segmentador como núcleo central del proceso de atención visual, en un esquema que será definitivamente presentado con el análisis de resultados del Capítulo 5. En este apartado, sin embargo, abordamos el estudio de mayor detalle de la implementación. Por ello, la Figura 4-1 muestra ahora un esquema de la arquitectura completa cuyo nivel de detalle es mucho mayor.

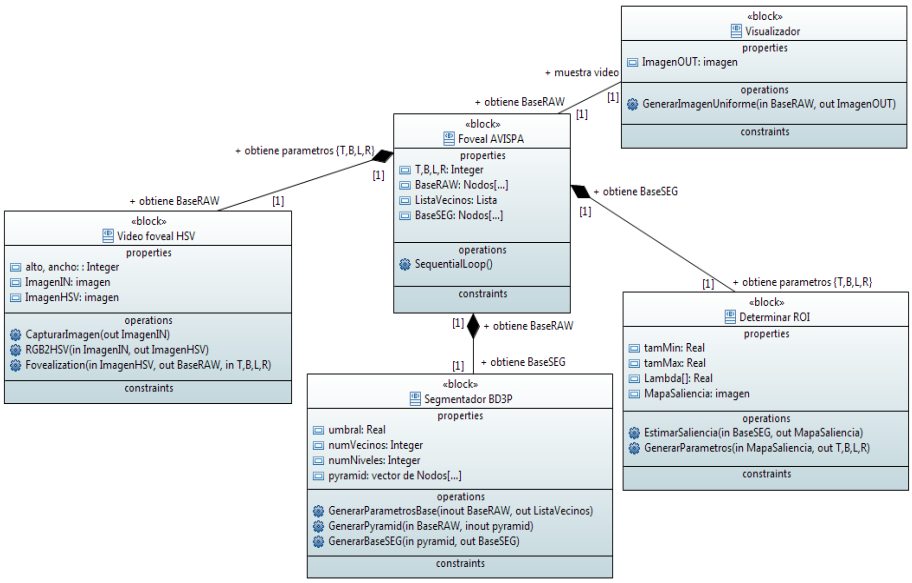


Figura 4-1: Arquitectura lógica del sistema implementado.

Básicamente, se puede asumir que la arquitectura del sistema completo (**`Foveal AVISPA`**) contiene tres bloques, encargados respectivamente de la captura de la imagen y generación de la base multirresolución (**`Video foveal HSV`**), la segmentación del grafo asociado a esta base (**`Segmentador BD3P`**), y la estimación de la nueva posición de la fóvea en función de una estimación de saliencia que se hace usando, como elemento básico, el grafo segmentado asociado a la base (**`Determinar ROI`**). Adicionalmente, la arquitectura cuenta con un cuarto bloque (**`Visualizar`**), que permite la monitorización del trabajo realizado mostrando las imágenes multirresolución, en crudo o segmentadas, por un monitor conectado al entorno de desarrollo. Las relaciones que muestran el diagrama SysML de la Figura 4-1 informan sobre un continuo trasiego de información entre los bloques de la arquitectura. Se puede trabajar con dos lazos cerrados. En el global y completo, que forman **`Video foveal HSV – Segmentador BD3P – Determinar ROI`**, el primero de los bloques tiene como salida el grafo multirresolución de la base de la pirámide irregular foveal, **`BaseRAW`**, y, como entrada, los parámetros que codifican la retinotopología (los parámetros **`{T, B, L, R}`**, pues el número de anillos y las dimensiones de la imagen, **`alto`** y **`ancho`**, se consideran constantes). El bloque **`Segmentador BD3P`** recibe el grafo **`BaseRAW`** y construye la pirámide irregular, **`pyramid`**. Se incluye en este bloque, por razones que se analizarán

en el siguiente párrafo, el cálculo de los vecinos de cada nodo de **BaseRAW** (**GenerarParametrosBase**), tanto la lista acotada que se usará en el BD3P como la lista total (**ListaVecinos**) -sólo para la base- que será empleada por el bloque **Determinar ROI** para el cálculos de los contrastes color e intensidad. En el nivel superior de esta pirámide se tendrá un grafo cuyos nodos serán las raíces de las clases en la imagen. Su propagación, por los enlaces entre-nivel, hasta la base permitirá definir la segmentación de la imagen (**BaseSEG**). Este grafo será la entrada del tercero de los bloques, que lo empleará para estimar las características de evidencia de cada región y estimar el mapa de saliencia, **MapaSaliencia.** La región más relevante será automáticamente seleccionada como el foco de atención y se estimarán los parámetros **{T, B, L, R}** de la nueva fovealización. Una segunda opción de trabajo es reducir el lazo a los bloques **Video foveal HSV – Segmentador BD3P**. El sistema capturará y segmentará imágenes con dimensiones y retinotopología fijas.

Todo este esquema ha sido implementado en Lenguaje C y evaluado en un procesador Intel® Core™ 2 Duo T8100 @ 2.10GHz proporcionando resultados correctos, pero con un tiempo de trabajo superior al segundo. Dejando fuera de su ámbito la referida actualización de las listas de vecinos, el primero de los bloques permite ser sintetizado en hardware en toda su totalidad. Por otra parte, la secuencialidad de las acciones que lleva a cabo con los datos el segundo de los bloques, **Segmentador BD3P**, permitiría organizar su trabajo usando los dos cores del procesador ARM del XC7Z020-CLG484-1. Básicamente la idea sería replicar el software en ambos cores pero procesar con ellos distintos fotogramas, simultaneando en el proceso el trabajo de los dos cores software, el core lógico de captura, y la entrada y salida de datos por los cores AXI *Video Direct Memory Access* (AXI VDMA). Este aspecto se verá con detalle en el apartado 4.3.

4.2. Implementación del bloque de adquisición de vídeo

Cuando hay que pasar de los modelos teóricos a su plasmación en plataformas reales nos encontramos con restricciones que están muy estrechamente relacionadas con el estado de la tecnología disponible en cada momento. En unos casos la tecnología nos condiciona qué es lo que podemos hacer y en cierta manera nos define la metodología de trabajo, basada en las herramientas de desarrollo disponibles. Así las contribuciones relacionadas con este trabajo, siempre han propuesto procesado de imágenes en escala de

grises y con dimensiones relativamente reducidas porque era la información que nos suministraba los sensores CMOS disponibles. En la parte de captura y preprocesado de la imagen, la capacidad de integración de los dispositivos programables nos limitaba el número de etapas que podemos integrar en el mismo dispositivo, obligándonos a una integración a nivel de sistema basado en varios circuitos, añadiendo la necesidad de diseño y desarrollo de interfaces a medida. Varios de los desarrollos recopilados en dichas contribuciones se realizan en entornos basados en captura de esquemas y síntesis lógica a partir de descripciones RTL, pues las herramientas de síntesis de alto nivel no están aún muy extendidas, ni disponibles a costes razonables. En cuanto a las alternativas de implementación disponibles, todavía se podía considerar el diseño y fabricación de un ASIC como una opción económicamente viable, dando pie a una línea de trabajo que incluye el diseño de un chip a medida para tareas de segmentación así como una metodología para la creación de vectores de prueba de chip fabrcado, de manera optimizada.

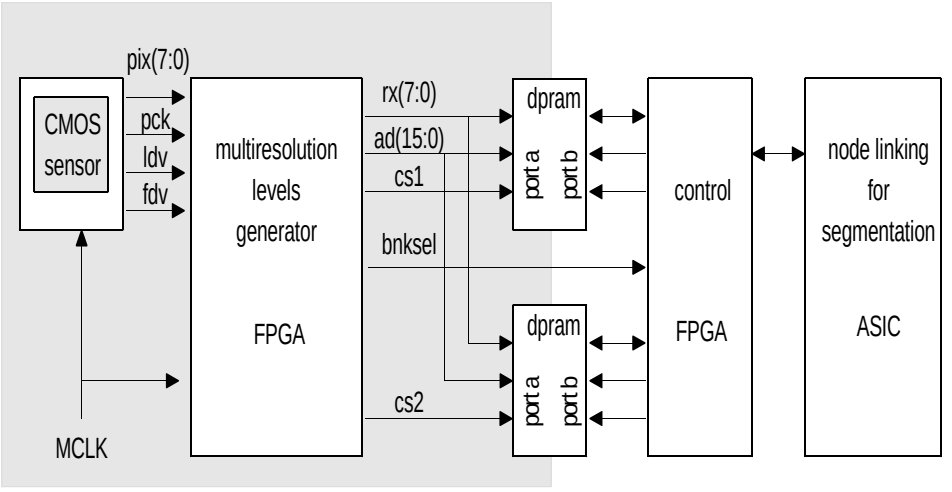


Figura 4-2: Diagrama de bloques de la integración del generador de niveles multiresolución, con los circuitos externos de procesado.

La Figura 4-2 muestra la integración a nivel del sistema del generador de niveles multirresolución presentado en (González, 2002) con el resto de la cadena de procesado, diseñada e implementada en un ASIC fruto del trabajo de (Coslado, 2004). En este diagrama hay que resaltar la necesidad de definir un interfaz a medida entre ambos subsistemas, basado en una arquitectura de

doble buffer utilizando memorias de doble puerto, con objeto de simplificar el acceso a ese espacio de memoria común. Este acceso hace necesario la integración, junto con el circuito generador de niveles multirresolución, de una etapa de codificación de prioridad, para realizar de forma adecuada la escritura de los diferentes niveles en dichas memorias, como se puede ver en la Figura 4-3.

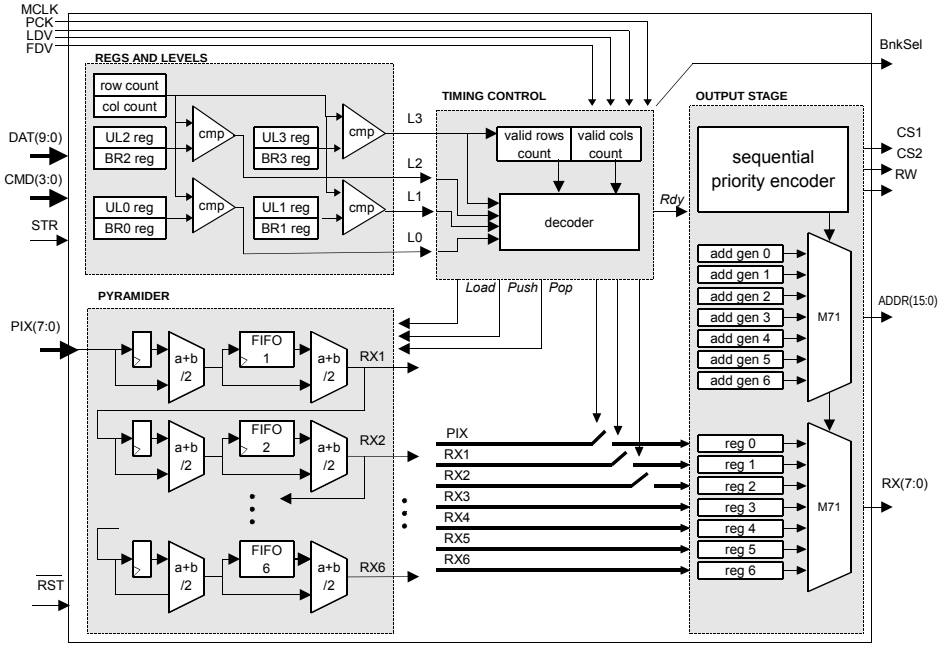


Figura 4-3: Diagrama de bloques del generador de niveles multiresolución que incluye el generador propiamente dicho y el interfaz de acceso a las memorias externas.

La evolución de la tecnología nos ofrece hoy en día muchas ventajas en varios de los aspectos estrechamente relacionados con el presente trabajo. En el caso de los sensores de imagen, disponemos a costes muy razonables de sensores CMOS, de calidad equiparable a los CCD, que nos ofrecen altas resoluciones en color, lo que nos permite seguir una línea novedosa con respecto a las contribuciones iniciales. Los dispositivos configurables ofrecen capacidades de integración que nos abren la posibilidad de integrar toda la cadena de procesado en un sólo dispositivo. Entre estos dispositivos configurables disponemos de auténticos SoC, referidos durante todo este documento como AP SoC, que integran en el mismo dispositivo tanto la capacidad de implementar procesadores hardware a medida como la de

ejecutar aplicaciones software sobre una unidad de procesado, todo ello integrado gracias a una arquitectura de bus perfectamente definida (ver Figura 4-4), lo que nos libera de la tarea de definir nuevos interfaces de conexión entre la parte hardware y software.

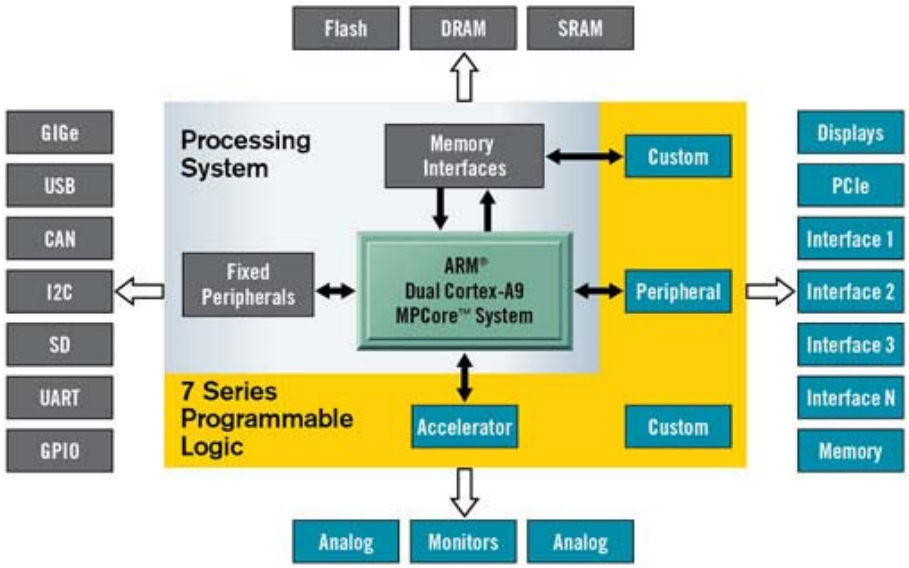


Figura 4-4: Arquitectura simplificada de un AP SoC difereciando la parte de procesado software (PS) y la de desarrollo de hardware a medida (PL), unidas mediantes un bus optimizado.

Esta plataforma hardware está acompañada de un entorno de desarrollo en el que, entre otras herramientas, disponemos de una de Síntesis de Alto Nivel (de aquí en adelante referida como HLS) que nos permitirá describir nuestros algoritmos en lenguaje C/C++, acelerando el proceso de diseño y verificación de los mismos. La transferencia de datos de video se integra en al arquitectura de bus gracias al estándar AXI4-Stream. El desarrollo de nuestros módulos teniendo en cuenta este marco, simplificará su integración a nivel de sistema.

En este apartado se describe el detalle del diseño de cada uno de las etapas que componen la cadena de preprocesado, tal y como se ilustra en la Figura 4-5, hasta entregar la información a la etapa de procesado cuyo detalle se desarrolla en el Apartado 4.3. Toda la cadena de captura de imágenes del sensor así como su prepocesado, son implementadas en la parte Programmable Logic (de aquí en adelante la parte PL), que en la Figura 4-4 se corresponde con toda la zona amarilla, donde se encuentran los recursos para nuestro hardware a medida

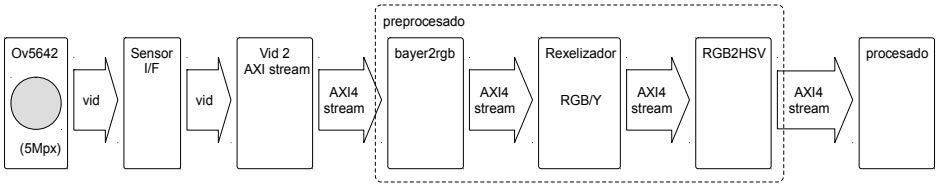


Figura 4-5: Cadena de preprocesado desde el sensor hasta la etapa de procesado.

Antes de abordar el diseño propiamente dicho, se revisan los aspectos importantes en relación con las estructuras de datos necesarias para los algoritmos de procesado de imagen y de video y los recursos de memoria disponibles para implementar dichas estructuras en la lógica configurable. Estos conocimientos nos permitirán orientar todo el diseño a un resultado óptimo en cuanto a prestaciones y utilización de recursos. Todo el desarrollo se realiza en C/C++ para ser sintetizado con HLS, por lo que es importante revisar las directivas necesarias para controlar el proceso de síntesis, en relación con el uso de los recursos de memoria.

4.2.1. Procesado de imagen y procesado de video

La forma en que están disponibles los datos que debemos procesar, tanto espacial como temporalmente, condiciona fuertemente el diseño de nuestro algoritmo así como de las estructuras de datos sobre las que vamos a trabajar.

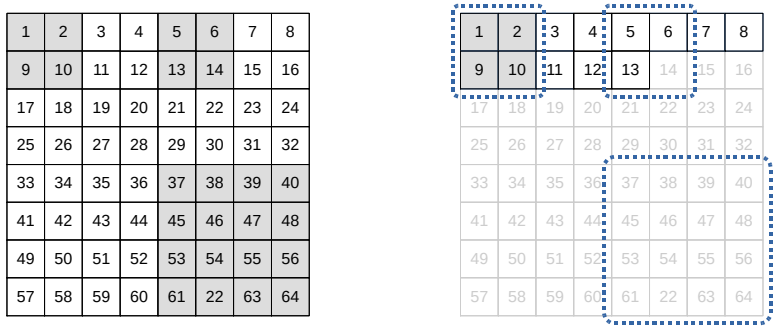


Figura 4-6: Distribución de los datos a procesar en una imagen estática y en una secuencia de video. Las celdas grises indican datos listos para ser procesados.

Nuestros algoritmos de procesado software pueden trabajar sobre los datos que componen una imagen estática, que están organizados como un array bidimensional y al que podemos acceder controlando los índices y el rango de

los mismos, para trabajar con cualquier zona de dicha imagen. Los datos están siempre disponibles, lo que nos permite en cualquier momento seleccionar la zona de trabajo o *ventana de procesado*, con las dimensiones que se necesiten (en la Figura 4-6 de la izquierda, zonas de 2x2 ó 4x4), y en el orden que se desee, pudiendo empezar por la parte superior de la imagen, bien por la izquierda bien por la derecha, o empezar por la parte inferior de la misma.

Cuando la imagen a procesar proviene directamente de un sensor que nos entrega los datos de forma progresiva, las ventanas de procesado van a estar disponibles con todos sus datos, en un orden y en un instante muy definido. En la Figura 4-6 de la derecha, podemos ver que las ventanas de procesado y la generación de los resultados asociados a ellas, van a estar disponibles progresivamente, desde arriba hacia abajo y de izquierda a derecha. La primera ventana destacada, pudo ser procesada cuando se dispuso de los datos número 9 y 10, que pertenecen ya a la segunda fila. La segunda ventana resaltada, no podrá ser procesada hasta que no se disponga del dato número 14. Para procesar la última ventana resaltada, habrá que esperar a tener datos que pertenecen al final de la imagen.

Los sensores de imagen entregan la información en un flujo o *streaming* de datos continuo, por lo que, una vez entregado, un dato no vuelve a estar disponible. Si tenemos en cuenta que los datos que completan una determinada ventana de procesado pertenecen a distintas filas, se hace necesario el diseño e implementación de estructuras de memoria locales que nos permitan almacenar toda esa información que ha sido recibida, a la espera de la llegada de los datos que nos interesan. Así, volviendo al ejemplo que nos muestra la Figura 4-6 de la derecha, debemos almacenar los datos del 3 al 8 pertenecientes a la primera fila, para poder completar la ventana con los datos 9 y 10 que pertenecen a la segunda fila.

Podemos concluir que los bufferes de memoria son una pieza clave para los algoritmos de procesado de video. Estas estructuras nos permiten mantener los contextos espacial y temporal necesarios para que una aplicación pueda trabajar sobre el dato actual. Cuando nuestro objeto es implementar este procesamiento en una determinada plataforma hardware, estas estructuras tienen impacto en la latencia, rendimiento, complejidad del procesamiento, y sin olvidar el correcto funcionamiento.

4.2.2. Recursos de memoria en la FPGA

Los bufferes de memoria que en el apartado anterior se han presentado como pieza clave en la implementación de algoritmos de procesado de video en

particular, necesitan de la disponibilidad de unos recursos en la tecnología que vamos utilizar. Las diferentes generaciones de FPGA que han ido apareciendo en el mercado, han ofrecido esta capacidad de implementar bloques de memoria empotrada dentro del propio dispositivo. Las características de estos recursos definen la arquitectura de memoria de dicha FPGA. Conocer las posibilidades que ofrece esta arquitectura es importante para orientar nuestro modelo RTL o de alto nivel, con el fin de que la síntesis e implementación, ya no sólo sea óptima, sino simplemente posible.

Por otra parte, la cantidad de recursos disponibles nos definen el tipo de estructura de memoria que podemos diseñar en nuestro sistema. Esta capacidad de memoria depende mucho del tipo de recurso utilizado, dentro de los que disponemos en la tecnología seleccionada. La Tabla 4-1 recoge de forma resumida, la evolución de estos recursos en función de la familia de dispositivo, así como los tipos de memoria de los que disponemos dentro de estas arquitecturas. Un dato muy importante recopilado en esta tabla, es la máxima cantidad de memoria que podemos implementar dentro de la propia FPGA en función del tipo de recurso utilizado, que en cualquiera de los casos, están en unos órdenes de magnitud muy por debajo de las capacidades de almacenamiento que nos ofrecen los circuitos externos de memoria.

	FF		Max	RAM (LUT)		Max	BRAM (18Kb)		BRAM (36Kb)		Max
	min	max	(Kb)	min	max	(Kb)	min	max	min	max	(Kb)
Spartan-6	4800	184304	180	1200	5420	1355	12	268	-	-	4824
Artix-7	20800	269200	262	800	11550	2888	312	1440	156	720	25920
Virtex-4	12288	178176	174	6144	89088	1392	48	336	-	-	6048
Virtex-5	19200	207360	202	4800	51840	3520	64	576	32	288	10368
Virtex-6	93120	948480	926	4180	33120	8280	312	1440	156	720	25920
Virtex-7	728400	2443200	2385	27750	86200	21550	1590	5168	795	2584	93024
Zynq	35200	554800	541	5280	83220	5200	120	1530	60	755	27180

Tabla 4-1: Recursos de memoria disponibles en las últimas familias de FPGA de XILINX

4.2.2.1. Flip Flops

Es el elemento más básico para construir estructuras de memoria. Los ocho Flip-Flop disponibles en un mismo SLICE permiten implementar un registro de 8bit, y con varios SLICE podemos configurar el banco de registros con las

dimensiones necesarias en nuestro procesamiento. Ofrecen la máxima versatilidad y ancho de banda tanto en escritura como en lectura, pero cuando las dimensiones de estas memorias son de valores medio-alto, suponen un uso muy ineficiente de los recursos de la FPGA.

4.2.2.2. Memoria Distribuida

Los generadores de función (LUT) disponibles en los SLICEM pueden ser configurados como pequeños bloques de memoria de escritura síncrona y lectura asíncrona. Combinando las cuatro LUT disponibles dentro del mismo SLICEM, se pueden crear bloques con diferentes e interesantes configuraciones como se puede observar en la Tabla 4-2, de entre las que destaca las configuraciones multipuerto. Estas configuraciones multipuerto permiten aumentar el ancho de banda de las lecturas, a costa de un incremento notable en la cantidad de recursos utilizados: lo podemos ver en la tabla como, por ejemplo, una memoria de 64 palabras de cuádruple puerto requiere cuatro veces más recursos que una de simple puerto y la misma capacidad.

RAM	Descripción		Número de LUT
64 x 1S	Single port	1 puerto escritura/lectura	1
128 x 1S			2
256 x 1S			4
64 x 1D	Dual port	1 puerto escritura/lectura; 1 puerto de lectura	2
128 x 1D			4
64 x 3SDP	Simple dual port	1 puerto escritura; 1 puerto de lectura	4
32 x 6SDP			4
64 x 1Q	Quad port	1 puerto escritura/lectura; 3 puerto de lectura	4
32 x 2Q			4

Tabla 4-2: Diferentes configuraciones de memoria mediante la combinación de las 4 LUT disponibles dentro de un mismo SLICEM

4.2.2.3. Memoria Dedicada

Cuando se necesitan memorias de mayor tamaño del que reflejan los datos de la Tabla 4-2, sería necesario combinar los recursos de un SLICEM con los de otros SLICEM, y esto se implementa de forma ineficiente. Así, cuando los requisitos de memoria son más importantes, es necesario emplear los bloques de memoria dedicada denominados BRAM (ver Figura 4-7). A diferencia de la

RAM distribuida, la capacidad es mucho mayor y tanto la escritura como la lectura son síncronas.

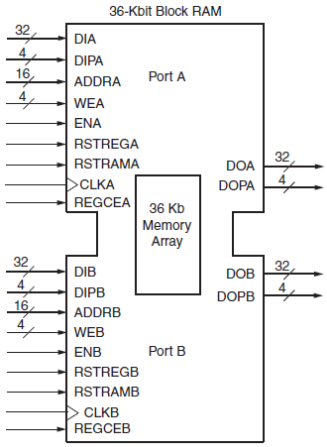


Figura 4-7: Bloque de memoria de 36Kb con 2 puertos de escritura/lectura y array de memoria compartida. Puede ser configurado como dos bloques de memoria de doble puerto de 18Kb cada uno, totalmente independientes.

Aunque el número de puertos disponible se limita a un máximo de dos, la anchura de datos de cada uno de ellos si es configurable de manera muy interesante tal y como se resume en laTabla 3. Estas posibles configuraciones permiten adaptar los bloques de RAM utilizados a la anchura de datos de nuestras unidades de procesamiento, es por lo tanto muy deseable ajustar ambos parámetros para un óptimo aprovechamiento de los recursos disponibles.

Modo de configuración	BRAM 18Kbit	BRAM 36Kbit	Descripción
Single Port	16Kx1, 8Kx2, 4Kx4, 2Kx9, 1Kx18	32k x 1, 16Kx2, 8Kx4, 4Kx9, 2Kx18, 1Kx36	1 puerto de escritura/lectura Lectura O escritura en 1 ciclo
True Dual Port	16Kx1, 8Kx2, 4Kx4, 2Kx9, 1Kx18	32Kx1, 16Kx2, 8Kx4, 4Kx9, 2Kx18, 1Kx36	Dos puertos de escritura/lectura totalmente independientes Dos operaciones en 1 ciclo
Simple Dual Port	16Kx1, 8Kx2, 4Kx4, 2Kx9, 1Kx18, 512x36	32K x 1, 16Kx2, 8Kx4, 4Kx9, 2Kx18, 1Kx36, 512x72	1 puerto de lectura y 1 puerto de escritura Lectura y escritura en 1 ciclo

Tabla 4-3: Posibles configuraciones de los bloques BRAM tanto en modos de funcionamiento como en la anchura de los buses de datos.

Podemos concluir de esta breve revisión de los recursos de memoria que el mayor ancho de banda se obtiene con estructuras basadas en Flip-Flop; que estructuras de datos importantes necesitan del uso de los BRAM; y que la

memoria distribuida es una solución eficiente y flexible cuando la cantidad de memoria requerida encaja dentro de los recursos disponibles en un mismo SLICEM.

4.2.2.4. Síntesis de Memoria

Los recursos para implementar estructuras de memoria son limitados y debe cuidarse su generación a partir de la síntesis, ya sea síntesis lógica o síntesis de alto nivel. En ambos tipos de síntesis, para crear estructuras de memoria hay que declarar arrays. En el caso de la síntesis lógica, la forma de actualizar y leer sus contenidos determina qué tipo de recurso, de los revisados anteriormente, se generará: si los contenidos son inicializables se crearán registros basados en Flip-Flop; si los contenidos no son inicializables y son leídos de manera asíncrona se crearán registros basados en LUT; y si los contenidos no son inicializables y son leídos de manera síncrona se utilizarán bloques de memoria dedicada o BRAM.

Cuando trabajamos con síntesis de alto nivel (HLS), no hay referencia explícita en el modelo a la señal de reloj, por lo que no se puede diferenciar entre accesos de tipo síncrono o asíncrono. HLS por defecto transforma los arrays utilizados en el modelo en BRAM. A menudo este funcionamiento es útil debido a que los bancos de registros son muy costosos, en términos Flip-Flop o LUT necesarios. No obstante las FPGA tienen un número bastante limitado de BRAM, como hemos podido ver en la Tabla 4-1. Por otro lado, los BRAM solo tienen dos puertos de acceso, lo que significa que, como máximo, solo se pueden realizar dos accesos simultáneos a la RAM, y esto puede limitar de manera seria el potencial paralelismo de nuestro diseño.

A diferencia de la síntesis lógica, la manera de controlar la generación de un tipo u otro de recursos de memoria está en el empleo de directivas de síntesis, que no forman parte el propio modelo desarrollado. La Tabla 4-4 recoge algunas de estas directivas relacionadas con los bloques de memoria.

Por ejemplo, si HLS utiliza un BRAM donde no nos interesa, aplicamos la directiva ARRAY_PARTITION con el tipo definido como COMPLETO y la dimensión fijada a 1, y forzaremos a que el array se implemente como registros, lo que supone un consumo importante de recursos pero nos ofrece un mayor paralelismo. Si deseamos ahorrar en numero de bloques BRAM utilizados, la directiva ARRAY_MAP nos permite unir múltiples arrays más pequeños en uno más grande, que pueda ser implementado en menos BRAM.

Directiva	Resultado obtenido
ARRAY_MAP	Agrupa múltiples arrays más pequeños en uno más grande, para ahorrar lógica de de acceso y bloques BRAM a costa de los tiempos de acceso
ARRAY_PARTITION	Divide un array grande en múltiples arrays más pequeños, permitiendo un incremento en el paralelismo en los accesos.
RESOURCE	Utilizado para especificar un recurso hardware concreto que debería ser utilizado para implementar un determinado elemento en el código fuente. Útil para especificar si un array debería ser implementado utilizando BRAM o utilizando LUT.

Tabla 4-4: Algunas directivas disponibles para controlar la generación de los distintos bloques de memoria durante el proceso de síntesis de alto nivel (HLS).

4.2.3. Estructuras de memoria para el procesado

De apartados anteriores hemos podido concluir que la memoria disponible en el interior de una FPGA es limitada y por lo tanto no es posible almacenar toda una imagen para su procesamiento. Es necesario implementar estructuras temporales, o bufferes de memoria de menor capacidad, que nos permitan procesar la imagen por partes. Tanto en aplicaciones de video como de DSP, los tres tipos de estructuras de memoria más comunes son los registros de desplazamiento, las ventanas de procesamiento y los bufferes de línea. Todas estas estructuras de memoria tienen consecuencias sobre la latencia, rendimiento y funcionalidad del hardware generado por las herramientas HLS.

En la Figura 4-8 se ilustra como las ventanas de procesado junto con los bufferes de línea, permiten “recorrer” la imagen completa para ser procesada, sin necesidad de tener toda esa imagen disponible en a memoria interna de la FPGA. Sin perder generalidad, podemos suponer que los píxeles que componen la imagen se suministran uno a uno, de forma progresiva, de izquierda a derecha y de arriba hacia abajo, siendo el primer píxel el de la esquina superior izquierda y el último el de la esquina inferior derecha.

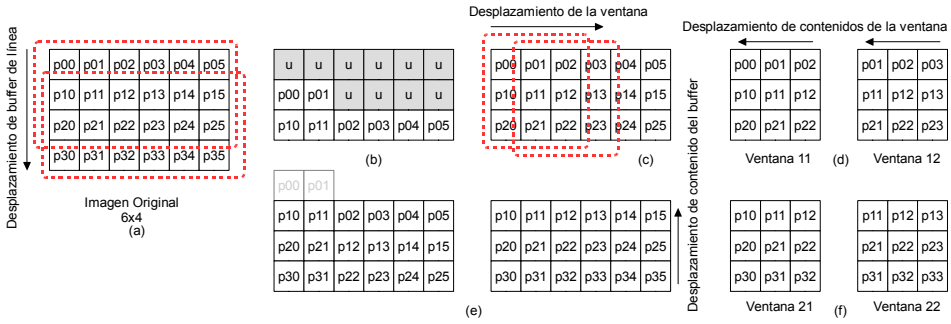


Figura 4-8: Estructura de los bufferes de línea y las ventanas de procesado

Los bufferes de línea permiten almacenar temporalmente el contenido de una o más líneas completas (3 en el ejemplo de la Figura 4-8) de la imagen original. El buffer de línea tiene por lo tanto una anchura igual a la anchura de la imagen original, y se desplaza verticalmente hacia abajo para “recorrer” toda la imagen (Figura 4-8a). Los píxeles de la imagen se van insertando en el buffer de línea por la parte inferior de la estructura, en el mismo orden que tienen en la línea original (Figura 4-8b). Esta inserción de datos supone un desplazamiento del contenido de cada columna del buffer de línea en sentido ascendente, por lo que el buffer se comporta como un array de registros de desplazamiento. En una primera aproximación, hasta que el buffer de línea no está completo (Figura 4-8c), no es procesado su contenido.

Generalmente el procesado de una imagen supone el procesado de un array de píxeles centrados en cada píxel a procesar. Este array es lo que conocemos como ventana de procesado y en el ejemplo de la Figura 4-8 es de dimensiones 3x3. Ya podemos ver una relación importante entre las dos estructuras: la altura del buffer de línea debe ser igual a la altura de la ventana de procesado. La ventana de procesado se desplaza de izquierda a derecha para recorrer todo el buffer de línea. Esto supone que los contenidos de la ventana se desplazan hacia la izquierda (Figura 4-8d) para permitir que los datos de cada columna del buffer de línea puedan ser insertados en la columna de la derecha de la ventana.

El desplazamiento hacia arriba de los contenidos del buffer de línea supone que cuando el buffer ya está completo, la introducción de nuevos datos por la parte inferior del buffer provoca el descarte de los datos que se encuentran en la parte superior del mismo (Figura 4-8e), reflejando una vez más el comportamiento de un registro de desplazamiento en cada columna.

4.2.3.1. Los Registros de Desplazamiento

Esta estructura básica no ha aparecido explícitamente como tal estructura, pero como ya hemos explicado, tanto el buffer de línea como la ventana de procesado, en definitiva son arrays de registros de desplazamiento. Su correcto modelado para su óptima implementación requieren una breve revisión.

Listado 1: modelado RTL de registro de desplazamiento

```
type int_arr is array (0 to 4) of INTEGER;
signal A: int_arr;
...
process begin wait until rising_edge(clk);
  for i in 0 to 3 loop
    A[i] <= A[i+1];
  end loop;
  A[4] <= new_data;
end process;
```

Listado 2: modelado HLS de registro de desplazamiento

```
int A[5];
int i;

for ( i = 0; i < 4; i++) {
#pragma HLS unroll
  A[i] = A[i+1]
}
A[4] <= new_data;
```

Tanto en el modelado RTL como en la herramienta HLS, la manera más simple de crear un registro de desplazamiento es la declaración de un array. En una descripción RTL (ver Listado 1), el comportamiento conocido de un registro de desplazamiento queda perfectamente definido sin ambigüedad alguna, de manera que en cada ciclo de reloj, el contenido del registro se desplaza, en este caso en sentido descendente de su índice, entrando por el extremo superior el nuevo dato a almacenar.

Sin embargo en HLS, no hay referencia explícita a la señal de reloj, pero a la hora de llevar a cabo la síntesis del modelo, cada una de las operaciones del bucle que muestra el Listado 2, se llevarán a cabo en ciclos de reloj distintos, necesitandose 4 ciclos de reloj para conseguir desplazar el contenido. Para conseguir que el resultado sea realmente un registro de desplazamiento, hay que dejar claro que todas las operaciones dentro del bucle se realicen simultáneamente. Para ello, el modelo mostrado utiliza la línea:

#pragma HLS unroll

que fuerza a la herramienta HLS a realizar todas las operaciones dentro del bucle en el mismo ciclo de reloj.

4.2.3.2. Las Ventanas de memoria

En procesado de imagen y de video, una ventana de procesado se define como un grupo de N píxeles centrados en un determinado píxel de la imagen. Como ya hemos visto, estas ventanas pueden verse como un conjunto de registros de desplazamiento el cual formaría un estructura de almacenamiento

bidimensional. Este tipo de memoria normalmente es implementada mediante el uso de Flip-Flop por dos razones:

- tiene pocos elementos, pues sólo los píxeles necesarios para calcular alguna característica del píxel central son almacenados. En el ejemplo de la Figura 4-8, la ventana es de dimensión 3x3 y necesita 9 elementos.
- Todos los píxeles que componen la ventana deben ser accesibles simultáneamente cuando se está haciendo el procesado correspondiente al píxel central.

La definición más básica de una ventana en C/C++ es un array bidimensional. Por ejemplo una ventana de dimensión 3x3 puede definirse como:

int B[3][3];

Una de las características de una ventana es que todos los elementos están disponibles simultáneamente, pero por defecto la herramienta HLS implementa este array como un bloque de RAM, con un número de puertos limitado como ya hemos revisado con anterioridad. La solución pasa por dividir el array en sus elementos individuales. En el caso de la variable B, esta división se consigue añadiendo a la declaración de la misma la siguiente línea:

#pragma HLS ARRAY_PARTITION variable=B complete dim=0

donde

- **complete**, divide una dimensión de un array en sus elementos individuales.
- **dim=0**, indica que esa división sea aplicada a todas las dimensiones en el array

Mover los datos en una ventana puede tratarse de la misma manera que un registro de desplazamiento. Recordemos que la ventana se desplaza hacia la derecha para “recorrer” todo el buffer de línea, por lo que los contenidos se desplazan hacia la izquierda al mismo tiempo que por la derecha se insertan los nuevos datos leidos del buffer de línea. Así, el movimiento horizontal de los datos se puede modelar tal y como se muestra en Listado 3.

Listado 3: modelado HLS del desplazamiento horizontal de los datos dentro de la ventana

```
for ( i = 0; i < 3; i++) {
#pragma HLS unroll
  B[i][0] = B[i][1]
  B[i][1] = B[i][2]
}
```

4.2.3.3. El buffer de línea

El buffer de línea es un registro de desplazamiento bidimensional que tiene capacidad para almacenar varias líneas de píxeles de la imagen a procesar. Normalmente los bufferes de línea se implementan internamente como bloques de RAM y evitar de esa manera la latencia de acceso a los circuitos de memoria externos. Por otro lado, son necesarios accesos simultáneos de escritura y lectura, lo que se hace posible gracias a la naturaleza multipuerto de los bloques de RAM disponibles. Aunque como hemos visto, una ventana de procesado es un subconjunto del buffer de línea, la mayoría de los algoritmos de procesado de imagen y de video no pueden trabajar directamente con un buffer de línea, por lo que sus contenidos deben ir pasando progresivamente a la ventana de procesado.

Como en el caso de una ventana, el buffer de línea se declara en C/C++ como un array bidimensional, teniendo en cuenta que dicho buffer ha de tener la misma altura que la ventana. En nuestro ejemplo, el buffer necesario para nuestra ventana B se declara como:

int D[3][MAX_COLS];

y la copia de datos del buffer a la ventana se realiza según el modelo que se muestra en el Listado 4.

Listado 4: modelado HLS de la inserción de datos delbuffer de línea en la ventana de procesado

```
for ( i = 0; i < 3; i++) {
#pragma HLS unroll
  B[i][2] = B[i][col]
}
```

Esta manera de declarar un buffer de línea es la más simple de codificar en C/C++, pero el declarar el buffer con N líneas completas no es lo más eficiente en término de uso recursos de la FPGA, ya que cada línea añadida requiere de un BRAM, pero los algoritmos de procesado no necesitan realmente disponer de las N líneas completas, sino que es suficiente con almacenar N-1 líneas y varios píxeles adicionales. Cuando la disponibilidad de recursos es un problema en nuestro diseño, deberíamos revisar nuestro modelo C del algoritmos para trabajar solo con N-1 líneas y así reducir el consumo de BRAM

4.2.4. Origen de la imagen a procesar

Los sensores CMOS comparten la misma tecnología de fabricación que los circuitos integrados analógicos y digitales necesarios para la captura, conversión, preprocesado, procesado digital de señal, adaptación de formatos,

interfaces de salida, así como el microcontrolador (MCU) necesario para gestionar todo este sistema integrado en un solo chip. Podemos hablar de que los sensores CMOS disponibles hoy en día son un claro ejemplo de System On Chip (SoC), ofreciendo al diseñador de sistemas un producto potente con una versatilidad fácil de controlar gracias a la disponibilidad de interfaces estandarizados y la integración de la MCU.

Sin buscar en la gama más alta de sensores disponibles, la siguiente Tabla 4-5 bien puede resumir las características principales de un sensor de resolución media-alta, con muy buenas prestaciones de frecuencia de refresco, y alta versatilidad en cuanto a formatos de salida.

Descripción	Resolución	Frame Rate	Formatos
5 megapixel	2592x1944	15 fps	YUV(422/420), YCbCr422, RGB565/555/444, CCIR656, 8/10-bit raw RGB data
1080p	1920x1080	30 fps	
720p	1280x720	60 fps	
VGA	640x480	60 fps	
QVGA	320x240	120 fps	

Tabla 4-5: Características principales del sensor de Omnivision ov5642

En la Figura 4-9 se muestra el diagrama de bloques de un sensor CMOS actual, que podría considerarse casi un diagrama genérico pues incluye los bloques básicos que ofrecen la práctica totalidad de los sensores que podemos encontrar en el mercado.

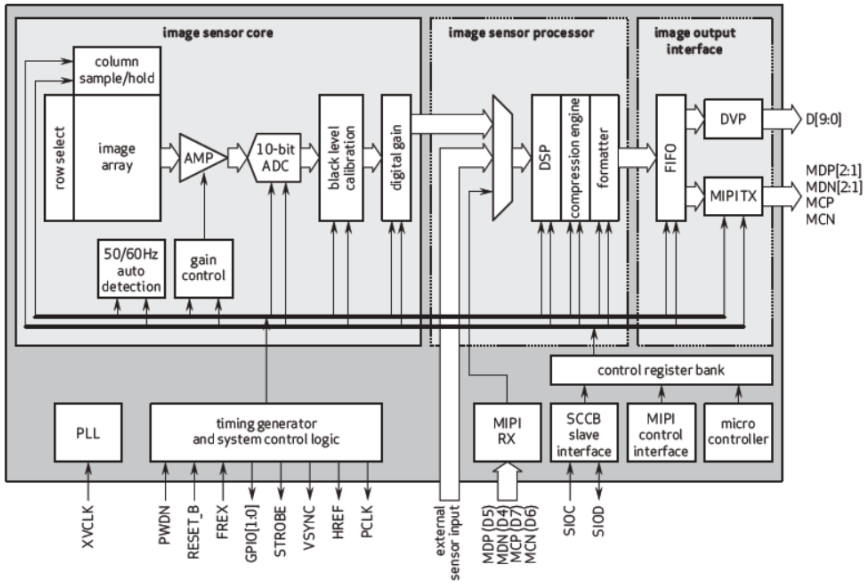


Figura 4-9: Diagrama de bloques de un sensor CMOS

En el diagrama debemos destacar que la información de cada pixel, a la salida del convertidor analógico/digital es de 10bit, resolución que se pierde en la etapa de compresión y formato, donde se generan y ofrecen los formatos habituales en la visualización de vídeo, tal y como aparecen recopilados en la Tabla 4-5 anterior. Esta opción es la más simple de manejar y la recomendada por el fabricante del sensor, pero si queremos trabajar con la máxima resolución tenemos la opción de trabajar directamente en el formato RAW RGB de 10bit, y realizar nuestro propio “revelado digital” de la imagen captada por el array del sensor, para reconstruir por interpolación la información completa de todos los componentes del color para cada pixel, ofreciendo al resto de la cadena de procesamiento un formato RGB888 (24 bit por pixel) o incluso un RGB101010 (30 bit por pixel) si fuera necesario. Esta opción también nos permitirá modificar y probar diferentes mecanismos de interpolación, y no estar sujetos al que implementa el propio sensor en su etapa de procesamiento digital de la señal (DSP), que por lo general suele ser el más simple en el que el valor de los colores que faltan en cada píxel toma el del píxel vecino del mismo color.

4.2.4.1. Temporización de las salidas de un sensor

A la hora de poder procesar la información que genera un sensor de video, tan importante son los propios datos que componen cada una de las imágenes,

como la señales de temporización que las acompañan. Como podemos ver en la Figura 4-10, la región activa es la zona donde se encuentran los píxeles que componen la imagen, y es la zona donde nuestro algoritmo debe trabajar. Las regiones de blanking, tanto horizontal como vertical, proporcionan el intervalo necesario para que los equipos de video detecten y visualicen correctamente estos contenidos.

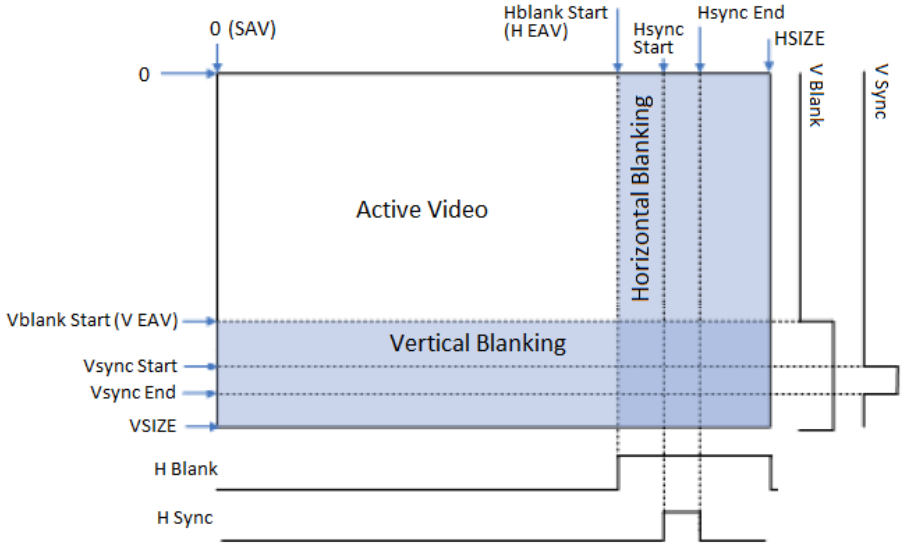


Figura 4-10: Formato básico de un frame de video que diferencia la región activa de las zonas de blanking, junto con las señales de sincronismo asociadas.

No todas las señales mostradas en la figura necesariamente han de ser generadas por el sensor. Así por ejemplo, podemos ver en el diagrama de bloques del sensor, que las señales de temporización generadas por el bloque correspondiente son solamente las del pulso de sincronización vertical (VSYNC) que indica la finalización de un frame y por lo tanto, el inicio de uno nuevo, y la señal HREF que no es otra que la señal complementaria de la de Blanking Horizontal (H Blank).

El aspecto que ofrecen las señales de sincronismo que entregan los sensores de imagen, se resumen en la Figura 4-11.

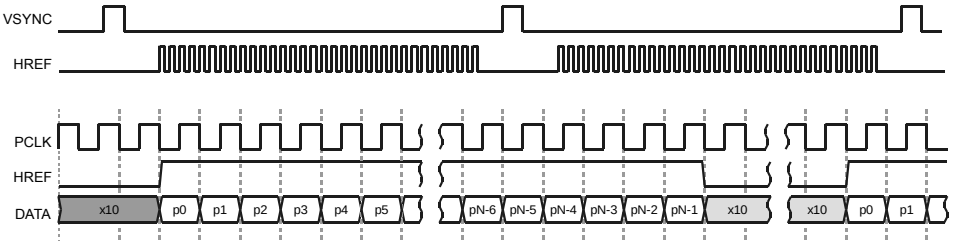


Figura 4-11: Señales de sincronismo asociadas al flujo de datos generado por un sensor de video.

En la parte superior de la Figura 4-11 se ilustra el sincronismo vertical, definido por la señal VSYNC, cuya frecuencia define los frames por segundo que está entregando el sensor. Cada frame está compuesto por tantas líneas como pulsos de la señal HREF hay entre dos pulsos consecutivos de de VSYNC. En la parte inferior se ilustra la temporización de línea, en la que podemos observar que solo se generan datos válidos mientras la señal HREF está activa. El resto de tiempo, que es el intervalo de blanking, no hay datos válidos, representado por un valor constante en el bus de datos (valor 0x10 en el ejemplo). Todas las señales están sincronizadas con la señal de reloj denominada PCLK, y los datos válidos se suceden al ritmo de un nuevo dato cada ciclo de PCLK. La frecuencia de PCLK fija el bitrate del flujo de datos y define un parámetro fundamental a la hora de modelar e implementar las distintas unidades de procesamiento que han de tratar los datos generados por el sensor. Esta frecuencia depende de la resolución y el el frame rate seleccionado en el sensor, como se puede ver en la Tabla 4-6:

Resolución	Frame Rate	Frecuencia PCLK
2592x1944	15 fps	80 MHz
1920x1080	30 fps	70 MHz
1280x720	60 fps	60 MHz
640x480	60 fps	20 MHz
320x240	120 fps	10 MHz

Tabla 4-6: Frecuencias de la señal de reloj de pixel (PCLK) en función de la resolución y el frame rate

Las señales de sincronización son fundamentales para dar formato apropiado al flujo de datos de video. Aunque no son el objeto principal de nuestros

algoritmos, estas señales deben ser adecuadamente interpretadas para que los resultados generados puedan ser procesados por otros bloques o equipos.

Si bien este esquema de temporización mostrado es el más habitual a la salida de nuestros sensores o a la entrada de los equipos de visualización, no es el único ni el más interesante a la hora de modelar nuestros algoritmos de procesamiento de video, que van a ser integrados junto con otros bloques dentro de un mismo sistema integrado en un solo chip. El esquema de temporización mostrado en la Figura 4-12, se centra en los propios datos que conforman la imagen a procesar, identificando claramente el primer dato o píxel de la imagen (con la generación del pulso SOF) así como el último píxel de cada una de las líneas de dicha imagen (con la generación del pulso EOL).

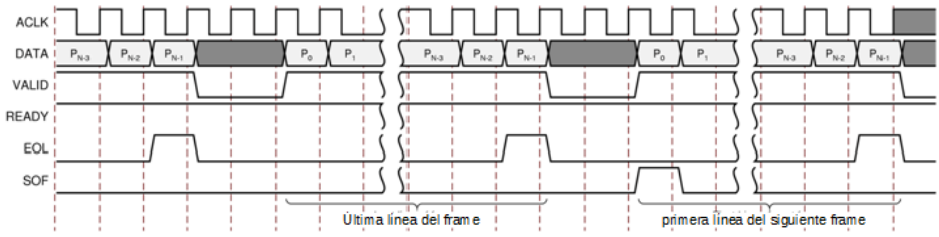


Figura 4-12: Esquema de temporización de un flujo de datos basado en la identificación del primer pixel de la imagen y del último pixel de cada línea.

Cuando trabajamos con HLS, debemos definir de manera explícita en nuestro modelo C/C++ cómo vamos a procesar las señales de sincronización del esquema de la Figura 4-11, puesto que esta herramienta es de propósito general y no tiene ningún conocimiento específico sobre procesado de video y en particular desconoce el significado de dichas señales de sincronismo. Esto hace que el modelo de nuestros algoritmos deba contemplar esta dificultad añadida, lo que dificulta el mantenimiento y modificación de nuestros algoritmos, además de que no nos sirva el esquema tradicional en procesado de video en términos de bucles sobre filas y columnas cuando utilizamos C/C++ para el modelado.

Xilinx suministra una solución a nivel de sistema que nos permite obviar el procesamiento de la señales de sincronización a la hora de hacer la captura de nuestro diseño para ser implementado mediante HLS. Esta solución es posible gracias a la utilización de dos IP cores: “Video In to AXI4-Stream” (PG043, 2014) que nos permiten transformar el flujo de datos según el esquema de la

Figura 4-11 en un flujo de datos según el esquema de la Figura 4-12; y "AXI4-Stream to Video Out" (PG044, 2014) que nos permite hacer la transformación en sentido contrario. Utilizando estos cores en nuestra cadena de procesamiento, podemos capturar nuestros algoritmos en términos de filas y columnas, tal y como se ilustra en el siguiente código:

Listado 5: Plantilla para la captura de algoritmos de procesado

```
void video_proc ( input_pixels, output_pixels, height, width ) {
  for (row=0; row<height; row++) {
    for (col=0; col<width; col++) {
      // algoritmo para el procesado de los pixeles
    }
  }
}
```

El código del Listado 5 muestra la plantilla recomendada y seguida en el presentre trabajo para modelar algoritmos de preprocesado de video con la herramienta HLS. A nivel de implementación hardware, la propuesta basada en estos cores y el uso de este esquema de codificación, resulta en el diagrama de bloques que se muestra en la Figura 4-13.

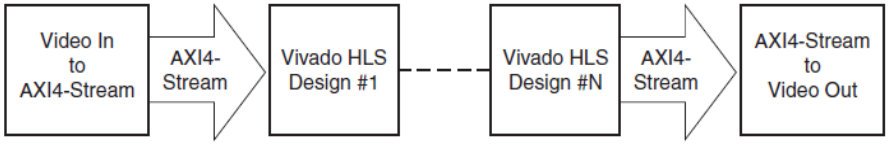


Figura 4-13: Diagrama de bloques de la cadena de procesamiento de video en Vivado HLS

En esta cadena de procesamiento, los dos IP cores aparecen al principio y al final de dicha cadena, y en la parte central se suceden las etapas de procesado creadas con la herramienta HLS y siguiendo la plantilla del Listado 5. Esta estructura y metodología es la que se ha seguido para crear y conectar las diferentes etapas de captura y preprocesado de la imagen suministrada por el sensor.

4.2.5. Interpolación de los valores del color

Tanto los sensores CCD como los sensores CMOS generan la información de color de sus píxeles haciendo uso de un filtro de color en forma de array, conocido como CFA o CFM (Color Filter Array o Color Filter Mosaic) de manera que cada píxel sólo tiene información de un color. Para obtener la información de los demás colores para cada píxel, es necesario un proceso de reconstrucción normalmente conocido como interpolación o *demosaicing*. Este proceso, en muchos casos, es una opción disponible dentro del mismo sensor, en una de sus etapas de procesado, pero también es cierto que hay sensores que no la integran y sólo tienen la opción de entregar los valores no reconstruidos para cada píxel: es lo que se conoce como formato RAW.

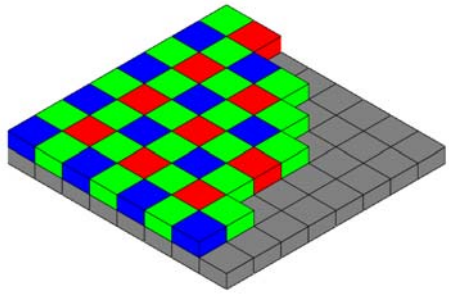


Figura 4-14: Distribución tipo Bayer de los filtros de color RGB sobre la matriz de un sensor de imagen

Aunque no es el único CFA utilizado, la mayoría de los sensores en la actualidad hacen uso del patrón Bayer (B.Bayer, 1976), cuyo aspecto es el de la Figura 4-14. En los últimos años han aparecido propuestas alternativas al clásico patrón Bayer, buscando mayor calidad en situaciones de captura de imagen exigentes. Concretamente el objetivo de estos últimos es ofrecer mayor brillo en capturas de alto contraste y un menor ruido en capturas en condiciones de baja luminosidad. Como se puede observar en la Tabla 4-7, en la matriz de filtros de color, algunas de las posiciones ocupadas por el filtro verde han sido reemplazadas por filtros transparentes, buscando una mayor captación de luz (Aptina, 2013).

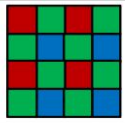
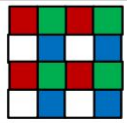
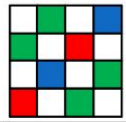
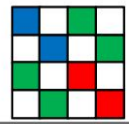
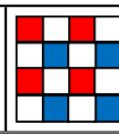
					
Pattern	**Bayer RG/GB**	**25%C**	**50%C: RG/BC Pattern A**	**50%C: RG/BC Pattern B**	**50%C: RC/CB Clarity+**
Unit cell	2 x 2	2 x 2	4 x 4	4 x 4	2 x 2
SNR improvement	0 dB (ref.)	1 dB	3-4 dB	3-4 dB	3-4 dB
Sharpness	reference	lower	slightly lower	slightly lower	equivalent
Spatial color artifacts	reference	slightly worse	serious	serious	equivalent

Tabla 4-7: Resumen de los nuevos CFA (Color Filter Array) propuestos comparados con el patrón Bayer como referencia.

A pesar de ciertas mejoras obtenidas por las nuevas propuestas, el patrón de Bayer sigue siendo el más empleado, y los métodos de interpolación basados en este patrón también han sido los más estudiados, desde métodos de interpolación lineal computacionalmente más simples (Sakamoto,1998) hasta propuestas mejoradas que requieren más capacidad de procesamiento (Ramanath, 2002; Malvar, 2004) .

4.2.5.1. Modelado e implementación del filtro de interpolación

Como ya hemos comentado anteriormente, los sensores comerciales ofrecen como método de interpolación el más simple de los conocidos, el del vecino más próximo, en el que cada pixel de salida interpolado se le asigna el valor del píxel más próximo en la imagen de entrada. Si hay varios vecinos con la misma distancia, se elige uno de ellos. Además, estos sensores no ofrecen a la salida el resultado completo de la interpolación, sino que muestran a la salida ese resultado comprimido en algunos de los formatos estándar que hemos revisado. Estas dos razones motivan el hecho de que se decida incluir como primer bloque de preprocesado, el filtro de interpolación que nos permitirá, como características principales, disponer de la máxima información de color para cada píxel, evitando así la etapa de compresión de datos, y la posibilidad de implementar y probar con diferentes métodos de interpolación.

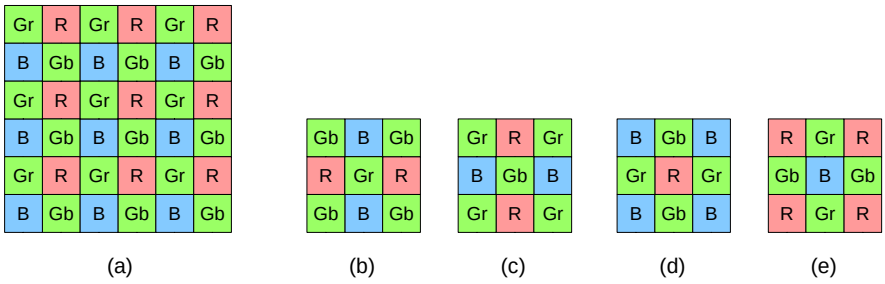


Figura 4-15: Patrón de Bayer para la captura de color junto con los filtros de interpolación bilineal comúnmente utilizados (Sakamoto, 1998) .

En la Figura 4-15a reproducimos el patrón de Bayer para facilitar la interpretación del método básico implementado en esta propuesta. Como se sugiere en (Sakamoto,1998), los valores de los componentes que faltan en cada pixel de entrada, son interpolados linealmente a partir de los valores de los píxeles vecinos del mismo color, teniendo en cuenta que la ventana de interpolación es una matriz de 3x3 centrada en el píxel que estamos procesando.

Para el cálculo de los componentes R y B, nos encontramos con cuatro posibles casos, ilustrados en la Figura 4-15b,c,d y e. Si R y B se calculan para un píxel tipo G, el promediado se hace con los dos píxeles más próximos del mismo color. Cuando el componente B se calcula para un pixel tipo R (Figura 4-15d), o bien el componente R se calcula para un píxel tipo B (Figura 4-15e) el promediado se realiza con los cuatro píxeles vecinos del tipo B y R respectivamente.

Para el cálculo del componente G, nos encontramos con dos posibles situaciones, ilustradas en la Figura 4-15d y e. En ambos casos el cálculo requiere el promediado de los 4 vecinos del tipo B.

El modelado de estos cálculos, para la posterior síntesis e implementación de los mismos, se ha realizado en base a la definición de distintos kernel de filtrado de tamaño 3x3, que se aplican de forma secuencial recorriendo la totalidad de la imagen a medida que es entregada por el sensor. De la explicación en los párrafos previos, se pueden determinar los cuatro kernel distintos que son necesarios y que se recogen en la Figura 4-16.

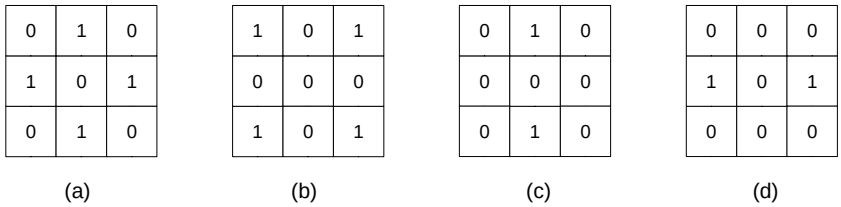


Figura 4-16: Matrices de interpolación para los distintos casos identificados: a) calculo de G en un pixel R ó B; b) cálculo de R en un pixel B, o cálculo de B en un pixel R; c) cáculo de B en un pixel Gr, o cácuo de R en un pixel Gb; d) cálculo de B en un píxel Gb o cálculo de R en un pixel Gr.

Estos kernel son aplicados de forma concurrente a los datos de entrada que se están procesando, de manera que los valores correctos para cada pixel procesado se seleccionan en función de la posición de dicho pixel en el patrón de Bayer. Como se observa, los kernel no incluyen el factor de división requerido en cada promediado (4 en los casos a y b; 2 en los casos c y d), con el objeto de poder trabajar con aritmética entera sin acumular el error por redondeo. El promediado se realiza en la última etapa del cálculo.

Siguiendo una vez más lo sugerido en (Sakamoto,1998), los resultados de la interpolación del color G, pueden mejorarse si se implementa una interpolación adaptativa que tenga en cuenta la correlación de valores de los píxeles vecinos. Como podemos observar en la Figura 4-17, la ventana de procesado pasa ahora a tener una dimensión de 5x5, y nos muestra los dos posibles casos que se nos puede presentar.

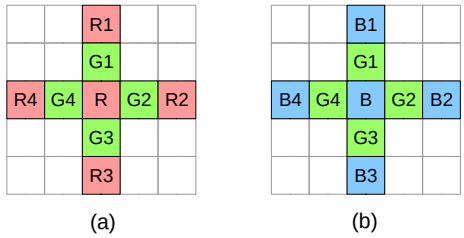


Figura 4-17: Ventanas de procesamiento para el cálculo del componente G mediante interpolación adaptativa.

En la Figura 4-17a, el valor del componente G debe ser calculado para un píxel de entrada de tipo R. Este valor viene dado por el siguiente promediado:

$$
G(R) = \begin{cases} (G1 + G3) / 2, & \text{si } |R1 - R3| < |R2 - R4| \\ (G2 + G4) / 2, & \text{si } |R1 - R3| > |R2 - R4| \\ (G1 + G2 + G3 + G4) / 4, & \text{si } |R1 - R3| = |R2 - R4| \end{cases}
$$

Es decir, se tiene en cuenta la correlación en el componente R para adaptar el método de interpolación. Si la diferencia entre R1 y R3 es menor que la diferencia entre R2 y R4, indicando que la correlación es mayor en sentido vertical, utilizamos la media de los vecinos verticales G1 y G3 para calcular el valor buscado. Si es la correlación horizontal la más fuerte, utilizamos los vecinos horizontales. Si ninguna de los correlaciones es dominante, utilizamos los cuatro vecinos para hacer el promediado.

De manera análoga, el componente G para un píxel de entrada del tipo B (Figura 4-17b), se calcula como:

$$
G(B) = \begin{cases} (G1 + G3) / 2, & \text{si } |B1 - B3| < |B2 - B4| \\ (G2 + G4) / 2, & \text{si } |B1 - B3| > |B2 - B4| \\ (G1 + G2 + G3 + G4) / 4, & \text{si } |B1 - B3| = |B2 - B4| \end{cases}
$$

La infraestrutura de este core basado en una ventana de procesado de dimensiones 5x5, puede ser utilizada para implementar otros algoritmos de reconstrucción bilineal que necesiten de este tamaño de ventana de procesado, como es el caso de la propuesta de (Malvar, 2004).

4.2.5.2. Generación del nivel de gris para cada pixel

En el apartado anterior se ha revisado con detalle cómo se obtienen los valores de los tres componentes del color para cada uno de los píxeles de la imagen. Podemos aprovechar que tenemos disponibles los valores calculados de estos componentes de color para, a partir de ellos, obtener el valor en escala de grises que corresponde a ese píxel. Siguiendo la recomendación de (ITU-R BT.601, 2013), el nivel de gris de un píxel se puede reconstruir aplicando la siguiente fórmula:

$$Y = 0{,}299R + 0{,}587G + 0{,}114B$$

El modelado directo de esta expresión es posible, pero significa la implementación de aritmética en punto flotante. Con objeto de operar con aritmética entera, y el ahorro en recursos lógicos que eso supone, la expresión

puede ser normalizada multiplicando ambos términos por un factor constante igual a 1024. La fórmula resultante sería:

$$Ynorm = Y * 1024 = 306\ R + 601\ G + 117\ B$$

Una vez realizado el cálculo, no debemos olvidar que la obtención del valor final requiere deshacer esta normalización:

$$Y = Ynorm / 1024$$

El utilizar una constante de normalización que es potencia de 2, nos garantiza que el resultado de la síntesis e implementación sea el más óptimo posible.

4.2.6. Generación de los niveles multirresolución

La estructura multirresolución se obtiene a partir partir de una imagen de resolución uniforme de dimensión H x V realizando un promediado 4 a 1, de manera que se obtiene una nueva imagen de dimensión (H/2) x (v/2). Un nuevo promediado 4 a 1 nos permite generar una nueva imagen de dimensión (H/4) x (V/4). El proceso se puede seguir repitiendo hasta obtener una imagen con dimensiones (H/2N) x (V/2N) donde N es el nivel de resolución reducida. La imagen original y las sucesivas imágenes de resolución reducida, definen un estructura piramidal tal y como se muestra en la Figura 4-18a, donde la base o nivel 0, es la imagen original.

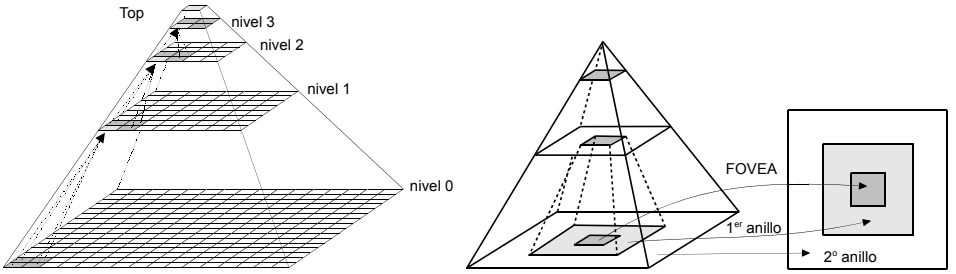


Figura 4-18: Estructura de la pirámide multirresolución su relación con la estructura foveal.

Para reducir la cantidad de información y así reducir el tiempo de procesamiento requerido, se puede seleccionar una región de interés (Region Of Interest – ROI) dentro de cada nivel de la pirámide. La ROI dentro del nivel 0 define la fóvea o zona de máxima resolución. Las ROI en sucesivos niveles cubren las ROI de niveles inferiores y añaden un nuevo anillo rectangular con menor resolución (Figura 4-18b). Se han propuesto múltiples configuraciones para definir estructuras foveales, desde fóvea centrada, pasando por la fóvea

desplazada y la fóvea adaptativa. Cada una de estas configuraciones requiere de sus propios parámetros para ser definida, pero en cualquiera de los casos enumerados, en una primera etapa siempre se hace necesario la generación de la estructura piramidal multirresolución, y después se pueden seleccionar las distintas ROI, simplemente definiendo sus límites dentro de cada nivel.

Es especialmente interesante cómo se pueden generar los datos correspondientes de los distintos niveles, para entender cómo se ha diseñado el datapath que ha de encargarse de dicha generación. Conviene recordar que los píxeles proceden del sensor en forma progresiva por lo que hay que contar con distintos recursos de almacenamiento temporal para poder completar los distintos promediados, por lo que como con cualquier procesado de imagen o de video, sería necesario la definición de un buffer de línea y la ventana de procesado de las dimensiones adecuadas. En este caso, puesto que el procesado consiste en un promediado 4 a 1, necesitaríamos una ventana de dimensiones 2x2, y un buffer de línea de dimensiones (2 x MAX_COL).

Teniendo en cuenta que el promediado 4 a 1 se puede realizar en dos etapas, promediando primero dos píxeles, los de la línea superior, y después los otros dos píxeles, los de la línea inferior, podemos proponer un procesado que no se base en el mecanismo clásico de ventana y buffer de línea, lo que a la hora de la fase de implementación supondrá un ahorro en recursos utilizados. La Figura 4-19 muestra el detalle de la generación de los niveles 1, 2 y 3 de la pirámide multirresolución, que nos permitirá analizar con detalle cómo se puede ir construyendo el resultado final. En dicha figura los instantes de interés, resaltados como celdas grises, están referidos a la temporización de la imagen original o nivel 0. Las etiquetas utilizadas deben ser interpretadas de la siguiente manera: *ldi* indica la captura del píxel del nivel i en un registro; *ui* indica el almacenamiento del primer promediado; *pi* indica la recuperación del primer promediado requerido para completar el segundo promediado; *rdi* indica el instante en el que el promediado global 4 a 1 está listo.

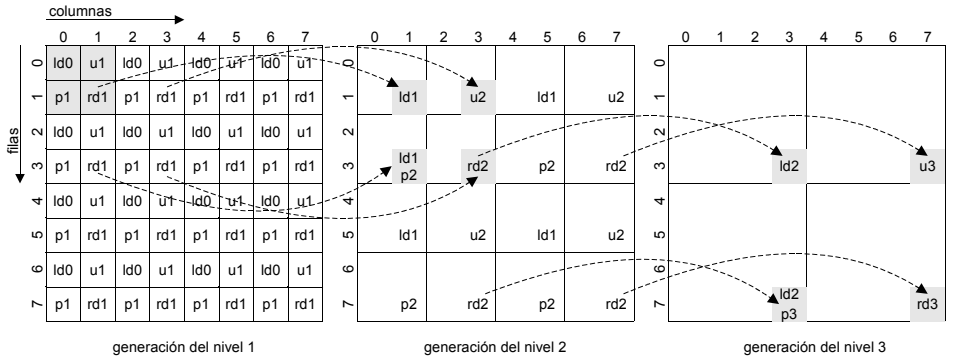


Figura 4-19: Generación de los valores correspondientes a los niveles 1, 2 y 3 de la pirámide multirresolución. El nivel 0, no mostrado aquí, representa la imagen original.

La imagen de la izquierda representa la generación del primer nivel. El píxel de la fila 0 y columna 0 debe ser registrado para ser promediado a continuación con el siguiente píxel de la misma fila. El resultado de este promedio está disponible en la columna 1 de la fila 0, de manera que en ese instante debe ser almacenado para posteriormente poder completar el promediado con los píxeles correspondientes de la fila 1. De igual manera se procede con todos los píxeles de la fila 0: se promedian dos a dos y se almacena el resultado. Trabajando de esta manera, podemos deducir que el buffer de almacenamiento necesario debe tener una dimensión igual a la mitad de la anchura de la imagen.

El píxel de la columna 0 y fila 1 debe ser registrado para ser promediado a continuación con el píxel de la columna 1, y este resultado debe a su vez ser promediado con el primer valor que almacenamos al procesar la fila 0, que debemos recuperar de dicho almacenamiento. De esta manera, el primer valor del nivel 1 estará disponible en la columna 1 y fila 1. De igual manera se procede con todos los píxeles que restan de la fila 1.

Podemos ver que en el buffer de almacenamiento introducimos valores durante la fila 0, y los extraemos durante la fila 1 en el mismo orden en que fueron almacenados, es decir, el buffer requerido es una FIFO.

Los valores del nivel 1 son generados en los instantes correspondientes con las columnas impares de las filas impares, las casillas etiquetadas en la Figura 4-19 como *rd1*, definiendo de esta manera un nuevo flujo de datos que se procesa de forma idéntica para generar los valores correspondientes al nivel 2 y así sucesivamente para los niveles superiores.

Con esta forma de proceder, el almacenamiento temporal requerido es de un solo registro y una estructura tipo FIFO con unas dimensiones de (1 x MAX_COLi/2), lo que supone un interesante ahorro de recursos respecto al uso de del buffer de línea de dimensión (2 x MAX_COLi) y la ventana de procesado de 2x2 que necesita de cuatro registros.

4.2.6.1. DataPath para la generación los niveles multirresolución

Cuando se diseña un datapath con el objetivo de su implementación hardware, es muy importante revisar la anchura de los datos procesados para acotar las dimensiones del datapath y controlar la cantidad de recursos necesarios. En este caso los datos a procesar requieren de aritmética entera y tienen una anchura bien definida de 8 o 10bit por cada canal de color. Se puede mantener esa anchura a lo largo del datapath, tal y como se ilustra en la Figura 4-20a, a expensas de los errores acumulados en los sucesivos promedios, o bien podemos adaptar de manera progresiva la anchura interna del datapath para representar adecuadamente el valor acumulado, y así evitar el desbordamiento, y dejar el promediado para una etapa final, tal y como se muestra en la Figura 4-20b.

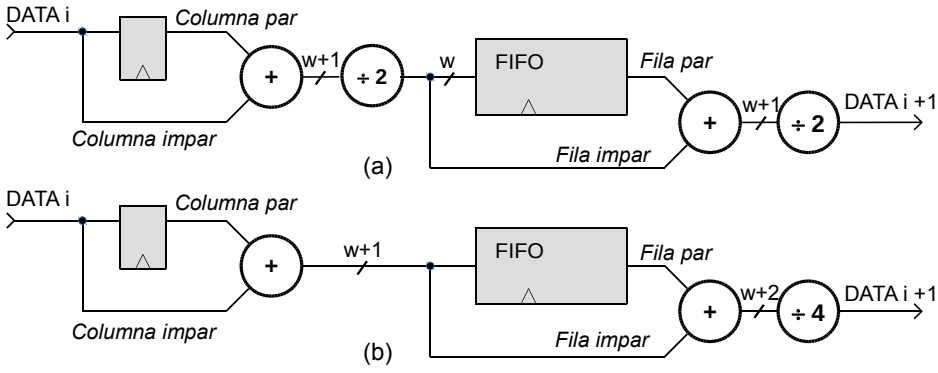


Figura 4-20: Estructura del DataPath necesario para la generación de los valores del nivel i+1 a partir del flujo de valores del nivel i.

Para generar la estructura piramidal completa necesitaremos tantas etapas, como la descrita en el párrafo anterior, como niveles multirresolución sean necesarios. La Figura 4-21 muestra la estructura completa necesaria para generar 3 niveles de resolución decreciente. En dicha figura destacamos cómo el promediado necesario en cada nivel se realiza en la etapa de salida, y como la anchura del datapath se va adaptando a medida que avanzamos en la cadena de procesado.

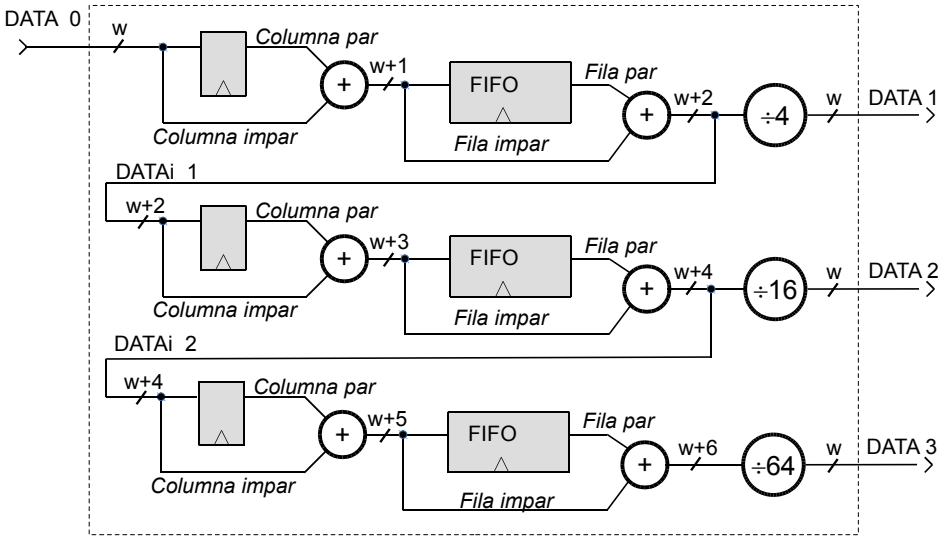


Figura 4-21: Estructura completa del datapath para la generación de todos los niveles (sólo se muestran los tres primeros) a partir de los datos del nivel 0.

Las opciones de posibles optimizaciones presentadas en este apartado para el modelado e implementación del datapath propuesto, cobra más importancia si se tiene en cuenta que hemos de procesar los tres canales de color, lo cual supone triplicar todas las estructuras analizadas y los recursos necesarios por cada una de ellas. Aunque el enfoque en la descripción esté más próximo a modelado del tipo RTL, es totalmente aplicable a descripciones de alto nivel en C/C++ para la herramienta HLS, tal y como así se ha hecho en el presente trabajo. Este punto incluye el control de la anchura del datapath, que en la herramienta HLS se consigue trabajando con los tipos de precisión arbitraria en lugar de los tipos estándar del modelado C/C++.

4.2.6.2. Generación de la imagen multiresolución

En los apartados previos se ha detallado cómo se generan los distintos niveles multiresolución que constituyen la estructura de una pirámide con diezmado regular. Pero de los capítulos previos hemos podido concluir que no necesitamos la estructura de dicha pirámide, ni siquiera la estructura del polígono foveal, pues el algoritmo propuesto trabaja de partida con una imagen multiresolución cuya retinotopología viene descrita, según la GMFD de fóvea de tamaño adaptativo, por los parámetros de subdivisión Ld, Rd, Td y Bd. Como podemos ver en la Figura 4-22, en la zona de la fóvea todos los píxeles son valores válidos y coinciden en esas coordenadas con los de la imagen original.

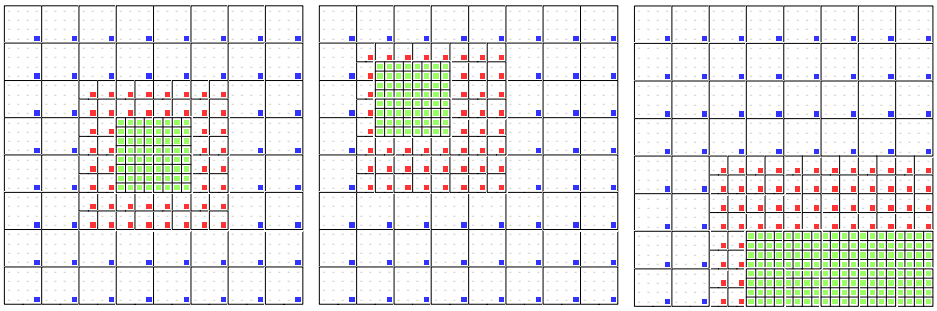


Figura 4-22: Diferentes retinotopologías en función de los valores de *Ld, Rd, Td y Bd,* para una estructura con dos anillos.

En los anillos de menor resolución, los valores originales que coinciden con esas coordenadas no tienen interés para nuestra imagen multiresolución (marcados en la figura con el carácter "-"). Sabemos que en el instante en que nos encontramos en la coordenada coincidente con el píxel de la esquina inferior derecha de cada uno de los réxeles, tenemos disponible el valor correspondiente al promedio de dicho réxel (son los puntos marcados con un cuadrito rojo en el anillo 1 y con un cuadrito azul en el anillo 2). De hecho, de la Figura 4-19, sabemos que, si por ejemplo nos encontramos en el anillo 5, en el mismo instante se tienen disponibles los promedios correspondientes a un rexel de cada uno de los 5 anillos además del píxel de la imagen original, pero sólo el valor correspondiente al anillo 5 es el que nos vale para nuestra imagen multiresolución.

Para una imagen de dimensiones HxW y una estructura de fóvea más 5 anillos, los límites del anillo *i-ésimo* vendrán dados por las siguientes coordenadas

$$xi_{min} = L_d \cdot \sum_{n=i+1}^{5} 2^n \quad , \quad yi_{min} = T_d \cdot \sum_{n=i+1}^{5} 2^n$$

$$xi_{max} = (W-1) - R_d \cdot \sum_{n=i+1}^{5} 2^n \quad , \quad yi_{max} = (H-1) - B_d \cdot \sum_{n=i+1}^{5} 2^n$$

donde $0 \le i < 5$, el anillo 0 se corresponde con la fóvea, y los límites del anillo 5 coinciden con los límites de la imagen. Además, un rexel sólo puede pertenecer a un sólo anillo.

4.2.7. Conversión entre los espacios RGB y HSV

El espacio de color RGB es el más adecuado para mostrar imágenes en dispositivos de visualización, además de ser el utilizado por los sensores cuando entregan las imágenes en modo RAW. Pero a la hora de realizar procesado de imagen y video no es el más indicado, al no tener las componentes de luminancia y crominancia por separado. Uno de los espacios de color ampliamente utilizado en tareas como segmentación, es el espacio HSV.

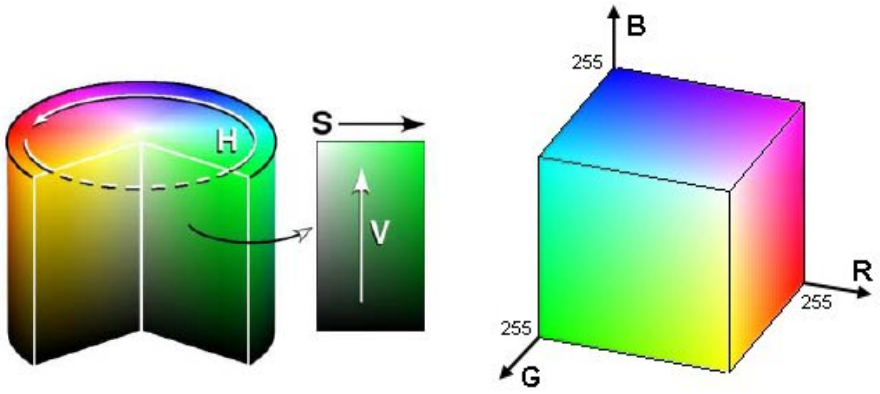


Figura 4-23: El espacio de color HSV puede ser representado por un cilindro y el espacio RGB como un cubo.

La conversión entre estos dos espacios de color se trata básicamente de una transformación entre coordenadas cartesianas y coordenadas cilíndricas. Su formulación está ampliamente documentada y es empleada en multitud de trabajos de procesado de imagen donde se pone énfasis en la necesidad de trabajar en el espacio HSV (Deswal, 2014, Chmelar, 2014), así como el interés en su implementación hardware (Hamachi, 2013).

La cadena de preprocesado propuesta en este trabajo incluye, en su última etapa, la conversión de los valores RGB a HSV, para su posterior utilización por los algoritmos de segmentación. Una vez realizado el procesado de la imagen, antes de poder realizar su visualización, es necesario incluir una etapa de conversión de los valores HSV a RGB.

Esta conversión sólo necesita los valores de un único píxel, por lo que no es necesario estructuras de almacenamiento temporal: ni ventanas de procesado ni bufferes de línea. Por lo tanto, la solución propuesta para su implementación

debe atender a la optimización en cuanto a los operadores aritméticos utilizados así como la resolución de dichas operaciones, sin comprometer la funcionalidad. Trabajar con aritmética en punto flotante nos ofrece la precisión necesaria, a costa de unos altos requerimientos de recursos de la FPGA, máxime teniendo en cuenta que la tecnología con la que trabajamos no dispone de unidades de procesamiento en punto flotante. La alternativa es utilizar la aritmética en punto fijo, con unos requerimientos de recursos menos exigentes pero a costa de una menor precisión.

4.2.7.1. Conversión de RGB a HSV

A partir de los valores normalizados R'=R/255, G'=G/255 y B'=B/255 podemos calcular los siguientes parámetros:

$$Cmax = \max (R', G', B')$$

$$Cmin = \min (R', G', B')$$

$$\Delta = Cmax - Cmin$$

Con los valores normalizados y los parámetros calculados, las ecuaciones que nos permiten realizar la transformación del espacio RGB al espacio HSV son las siguientes:

$$H = \begin{cases} 0 & , \text{si } \Delta = 0 \\ 60 \times ((G'-B') / \Delta \mod 6) & , \text{si } Cmax = R' \\ 60 \times ((B'-R') / \Delta + 2) & , \text{si } Cmax = G' \\ 60 \times ((R'-G') / \Delta + 4) & , \text{si } Cmax = B' \end{cases}$$

$$S = \begin{cases} 0 & , \text{si } Cmax=0 \\ \Delta / Cmax & , \text{si } Cmax \neq 0 \end{cases}$$

$$V = Cmax$$

donde los valores generados cumplen que $0 \leq H < 360$, $0 \leq S \leq 1$ y $0 \leq V \leq 1$.

Todas las operaciones que se emplean en las ecuaciones anteriores son sintetizables por la herramienta HLS, dependiendo la cantidad de recursos utilizados del tipo de datos que utilicemos.

4.2.7.2. Conversión de HSV a RGB

A partir de los valores H,S y V, la reconstrucción del valor RGB correspondiente viene definido por las siguientes ecuaciones:

$$C = V \times S$$

$$X = C \times (1 - |(H / 60°) \bmod 2 - 1|)$$

$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0° \leq H < 60° \\ (X, C, 0) & , 60° \leq H < 120° \\ (0, C, X) & , 120° \leq H < 180° \\ (0, X, C) & , 180° \leq H < 240° \\ (X, 0, C) & , 240° \leq H < 300° \\ (C, 0, X) & , 300° \leq H < 360° \end{cases}$$

$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

Al igual que en la otra conversión, todas las operaciones que se emplean en las ecuaciones anteriores son sintetizables por la herramienta HLS, dependiendo la cantidad de recursos utilizados del tipo de datos que utilicemos. Como ya hemos comentado, la aritmética en punto flotante está soportada por la herramienta de síntesis, ofreciendo resultados con la precisión requerida, siendo las dos conversiones completamente reversibles, permitiendo recuperar el valor en el espacio de color original. Pero cuando nuestro objetivo es la integración en una plataforma empotrada, habrá que tener en cuenta la transferencia de resultados generados a otras etapas de procesado, a través de los buses disponibles. Habrá que sopesar la viabilidad de trabajar con la máxima precisión posible manteniendo la conectividad.

4.3. La conexión entre la parte HW y la parte SW

Una vez descritos los detalles del diseño y desarrollo de los bloques que componen el subsistema que se ha implementado en la parte PL del AP SoC, que constituyen la cadena de preprocesado necesario para generar una imagen multiresolución sobre la que trabajará el algoritmo de segmentación, y antes de entrar en la descripción de los detalles del desarrollo softeware de dicho algoritmo, dedicamos este apartado a definir la arquitectura propuesta en esta Tesis para la integración de todo el sistema en el mismo chip. Como ya anticipamos, la clave está en el acceso compartido que, tanto hardware como

software, han de hacer a la misma memoria DDR donde se almacenan los sucesivos frames capturados y que ha de ser procesados.

4.3.1. La estructura de bloques del sistema

En la Figura 4-24 se muestra donde se encuentran encuadrados todos los bloques desarrollados como hardware a medida para la parte PL.

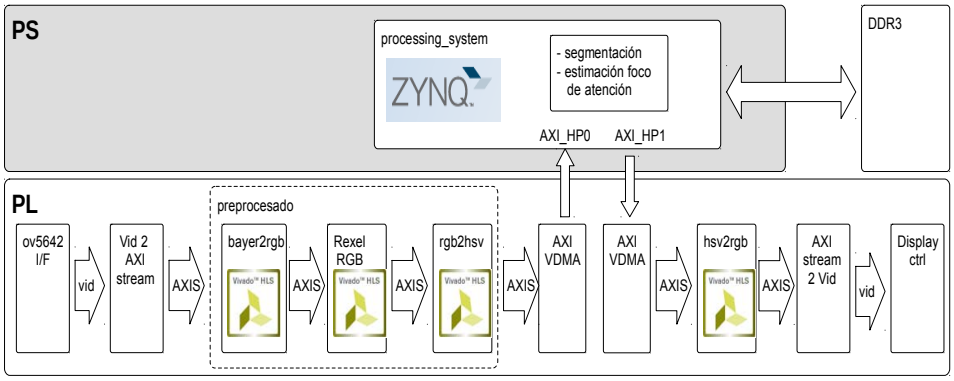


Figura 4-24: Diagrama de bloques de la distribución de los distintos procesos entre la parte PS y la parte PL del APSoC. A la DDR se accede desde el controlador asociado a la parte PS.

Las imágenes multiresolución generadas por la cadena de preprocesado se almacenan en DDR a través de los bloques AXI-VDMA y los canales de altas prestaciones AXI-HP. El procesado SW se realiza en la parte PS, y las imágenes resultantes de dicho procesado quedan almacenados en otra zona de la memoria DDR, de donde son recuperadas por otro bloque AXI-DMA a través de otro canal de altas prestaciones AXI-HP para pasarlas al canal de salida, donde se recupera el espacio de color RGB y ser visualizado en un dispositivo externo a través del controlador de display.

Como se observa en el diagrama se han utilizado dos ipcores AXI-VDMA independientes, uno para la entrada y otro para la salida de vídeo. Esto se he hecho así porque queremos que trabajen de forma independiente, y no queremos que se sincronicen internamente a través de los modos de “genlock”. Para un canal de vídeo normal sería deseable que el AXI-VDMA sea el mismo y que sus canales de entrada y de salida “se hablen” para que cada uno sepa qué frame está en uso por el otro.

4.3.2. La organización del espacio de memoria

En nuestra propuesta de funcionamiento, como queremos procesar el vídeo por software en el ARM de la parte PS, la gestión al acceso a la memoria será

diferente al de un canal de video normal: como podemos ver en la Figura 4-25 en la memoria DDR tendremos varios frames de vídeo definidos con la idea de que el software le diga al VDMA de entrada dónde debe guardar el video entrante, y al VDMA de salida de dónde leer la imagen que se está representando en el HDMI. Esto nos permite tener el control completo sobre el flujo de la imagen.

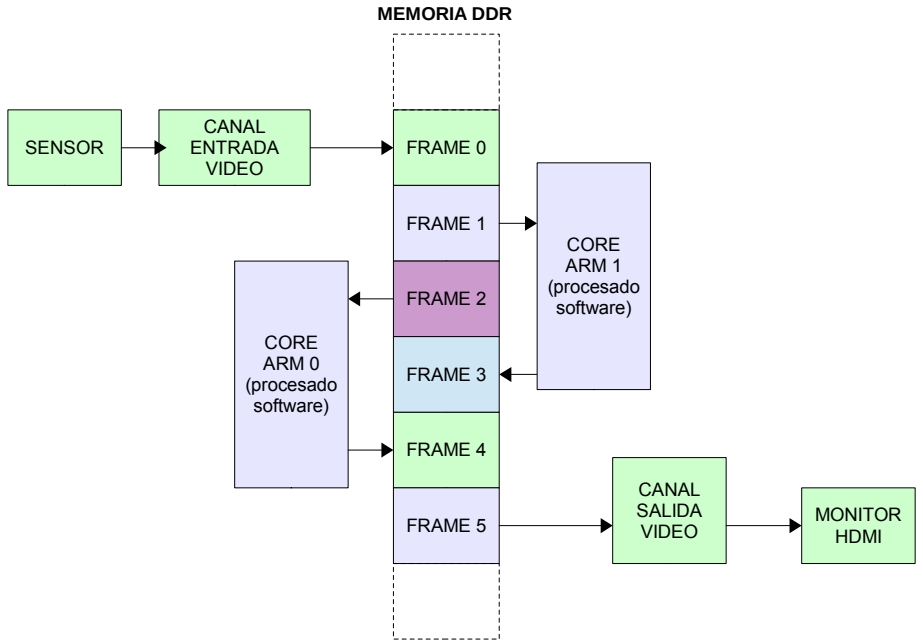


Figura 4-25: Segmentación del espacio de memoria para el procesado de los distintos frames

El funcionamiento, de manera muy esquematizada, sería el siguiente:

1. El canal de entrada de vídeo está volcando la imagen del sensor en el frame 0;
2. El core 0 del ARM está leyendo la imagen del frame 1 y generando el resultado de la imagen segmentada en el frame 3;
3. El core 1 del ARM está leyendo la imagen del frame 2 y generando el resultado de la imagen segmentada en el frame 4;
4. El canal de vídeo de salida está mostrando la imagen que hay en el frame 5 en el monitor por la salida HDMI.

Cuando uno de los cores del ARM, por ejemplo el Core 0, haya terminado su tarea de segmentación, se procederá como sigue:

1. Se le indica al canal de entrada de vídeo que el siguiente frame donde escribir es el frame 1;
2. Se le dice al canal de salida de vídeo que el siguiente frame de donde leer es el frame 3 (acaba de ser generado);
3. Se le dice al core 0 del ARM que ahora le toca leer la imagen del frame 0 y generar el resultado en el frame 5.

Con este esquema se consigue procesar de forma más fluida los frames procedentes del canal de video de entrada, y tenerlos disponibles para su visualización a través del canal de video de salida.

4.4. Implementación de los bloques de Segmentación y Estimación de la posición del Foco de Atención

Si el proceso de adquisición de la imagen, conversión de color y fovealizado han sido llevados con éxito a la parte lógica del AP SoC XC7Z020-CLG484-1, el algoritmo de segmentación y estimación de la nueva región de interés, los bloques **`Segmentador BD3P`** y **`Determinar ROI`** de la Figura 4.2, siguen una estrategia secuencial, más adecuada de ubicar en la parte software del AP SoC. El diagrama de secuencias en el que se gestiona la segmentación de la base multirresolución se presenta en la Figura 4-26. Básicamente, el principal consta de un bucle continuo de movimientos de fóvea, en el que se segmenta la imagen, se estima una nueva posición de la fóvea y se captura una nueva imagen. En concreto, la segmentación se inicia con la lectura del nuevo grafo, que almacena la imagen multirresolución como un vector (**`BaseRAW`**). Este vector unidimensional almacena el valor HSV de cada nodo del grafo, y el índice o posición en el vector nos indica su posición en la retinotopología (ver Capítulo 3). Sin embargo falta por completar la información referida a sus vecinos en la retinotopología (acotados a un número máximo y a ser similares en color). Esta información se completa usando la función **`GenerarParametrosBase`**. Simultáneamente, esta función completa información adicional sobre los nodos, necesaria para el cálculo de la saliencia. Esta información no se almacena en los nodos de **`BaseRAW`**, pues esto obligaría a definir una estructura muy pesada para el tipo nodo y, en realidad, estos datos solo se necesitan para la base o nivel 0 de la pirámide. Es por ello que se crea la estructura de almacenamiento **`ListaVecinos`**. Su uso quedará

relegada a las funciones **`EstimarSaliencia`**, ya en el bloque **`Determinar ROI`**.

Una vez se dispone del grafo asociado a la base, se procederá a generar la pirámide irregular foveal usando el algoritmo BD3P. La función **`GenerarPyramid`** consta de tres partes secuencialmente ejecutadas: la generación de cada uno de los niveles de la jerarquía (**`GenerarNivel`**), en un proceso ya descrito en el Capítulo 3, la asignación de un identificador único de clase a cada nodo superviviente del nivel superior de la jerarquía (**`ActualizarClase`**), y la transmisión, en un proceso *top-down* que emplea los enlaces inter-niveles creados por **`GenerarNivel`**, del identificador de Clase y del valor de color HSV desde las raíces o padres del nivel superior de la jerarquía hasta la base. Estos datos se actualizan en la pirámide creada, pero la base, representada ahora por un conjunto de regiones o clases (proto-objetos) se graba como **`BaseSEG`**.

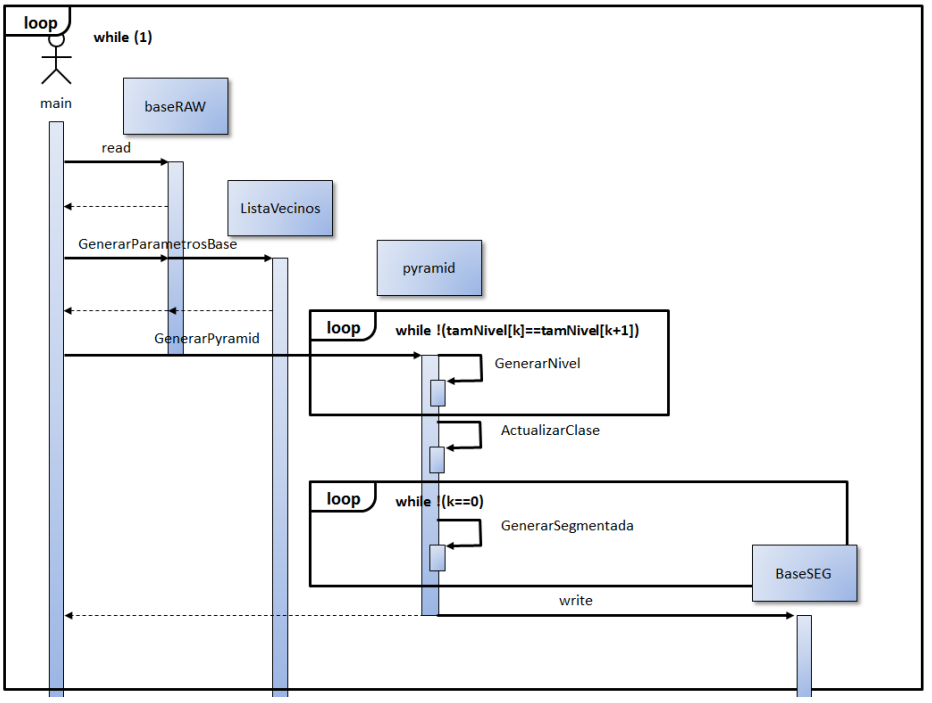


Figura 4-26: Diagrama de secuencia asociado al proceso de segmentación

Se aprecia el carácter secuencial del proceso de segmentación de la imagen multirresolución que, como se comentaba al inicio de este apartado, justifica su implementación en la parte software del AP SoC. Aparte del programa principal,

cuya actuación seguirá ahora con la estimación de la saliencia y la posición del nuevo foco de atención, sólo los datos almacenados en **`ListaVecinos`** y **`BaseSEG`** son necesarios pues permitirán la estimación de los mapas de evidencias y saliencia en el bloque **`Determinar ROI`**. El diagrama de secuencia de este proceso se muestra en la Figura 4-26.

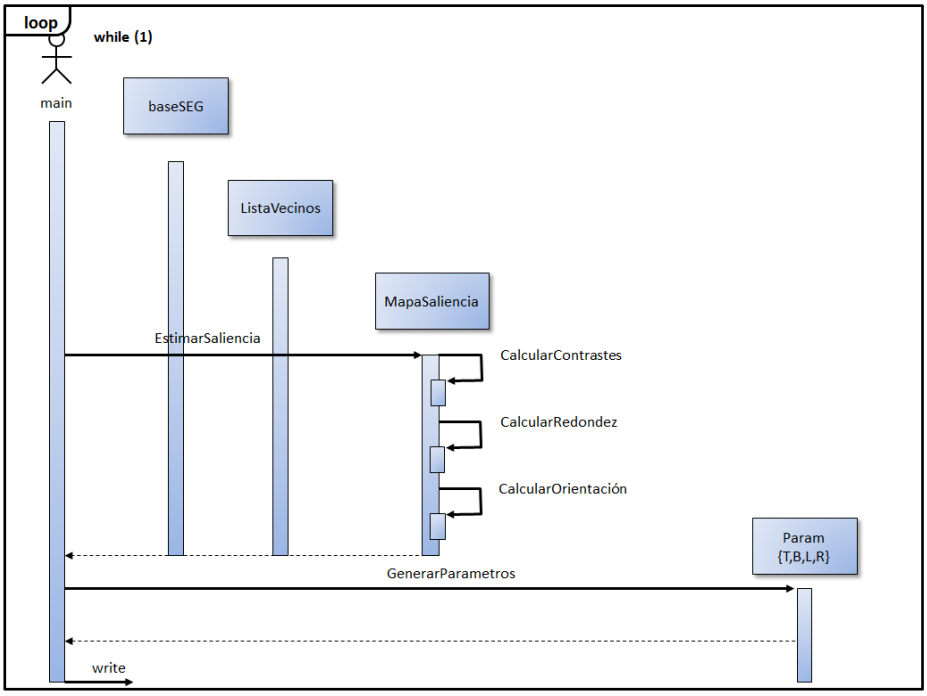


Figura 4-27: Diagrama de secuencia asociado al proceso de estimación de la saliencia y la posición del nuevo foco de atención

El proceso se fundamenta ahora en dos funciones principales: **`EstimarSaliencia`**, que emplea los datos en **`BaseSEG`** y **`ListaVecinos`** para computar los contrastes color e intensidad, la redondez y el contraste en orientación, y **`GenerarParametros`**, función con la que se cierra el lazo de control del bucle principal, y que estima los valores de los parámetros que definen la retinotopología foveal (**`T,B,L,R`**). Al actualizar estos parámetros, el hardware de adquisición de imagen modificará la posición de la fóvea y devolverá una **`BaseRAW`** actualizada. Se ha cerrado el bucle principal de funcionamiento.

En los siguientes apartados se proporciona información adicional sobre la implementación de estos dos bloques. Comentar que el bloque

`Visualizador`, cuya función es proporcionar un flujo de vídeo a través del conector HDMI de la placa Zedboard, puede intercalarse en cualquier posición de este bucle principal. Como se describió al inicio de este Capítulo, su funcionamiento se tratará de paralelizar con el de los cores principales, pues la generación de una imagen de gran tamaño (1920x1024 si se quiere usar todo el campo del HDMI) es costosa computacionalmente. En futuras modificaciones de la arquitectura propuesta, este bloque de verificación debería sintetizarse en hardware.

4.4.1. Detalles de implementación del algoritmo de segmentación

El primero de los problemas que se debe afrontar en el bloque **`Segmentador BD3P`** es que el esquema de codificación del grafo no permite una fácil determinación de los vecinos de cada nodo. Esto es, no es fácil conocer los identificadores de los nodos que rodean, en vecindad V-4, a cada nodo. Para resolverlo se recurre al empleo de dos ventanas auxiliares, que irán recorriendo la imagen multirresolución de arriba a abajo para determinar los vecinos de cada nodo. Una de esta ventanas sirve para almacenar toda una fila de posibles vecinos superiores. Su funcionamiento se ilustra en la Figura 4-28. En la esquina superior-izquierda se muestra una imagen multirresolución de dos anillos de resolución y fóvea de 4 x 6 píxeles. En la esquina superior-derecha de la figura se muestra, con un conjunto de flechas, las posiciones en las que se actualizan los valores de esta ventana. En concreto, cuando se almacenan los valores en la posición marcada como A, se almacenan dos valores por nodo. Eso es debido a que la siguiente fila corresponde al siguiente anillo de resolución, y el tamaño de rexel se reducirá a la mitad. Por ejemplo, en el caso del rexel marcado en color verde, se actualizan con el identificador de nodo del rexel las dos posiciones marcadas en verde de la ventana. Los dos réxeles del siguiente anillo de resolución, justamente bajo el rexel verde, podrán ahora comprobar en esta ventana si el rexel verde es su vecino por similitud en color. En **`ListaVecinos`**, sin embargo, estas vecindades se almacenarán directamente, al no imponerse restricción por parecido en color. En la posición marcada con B (esquina inferior-derecha), la ventana guarda el identificador de los nodos marcados en rojo y azul. El rexel inmediatamente inferior, de tamaño doble al estar en el siguiente anillo de resolución, puede tener dos vecinos superiores.

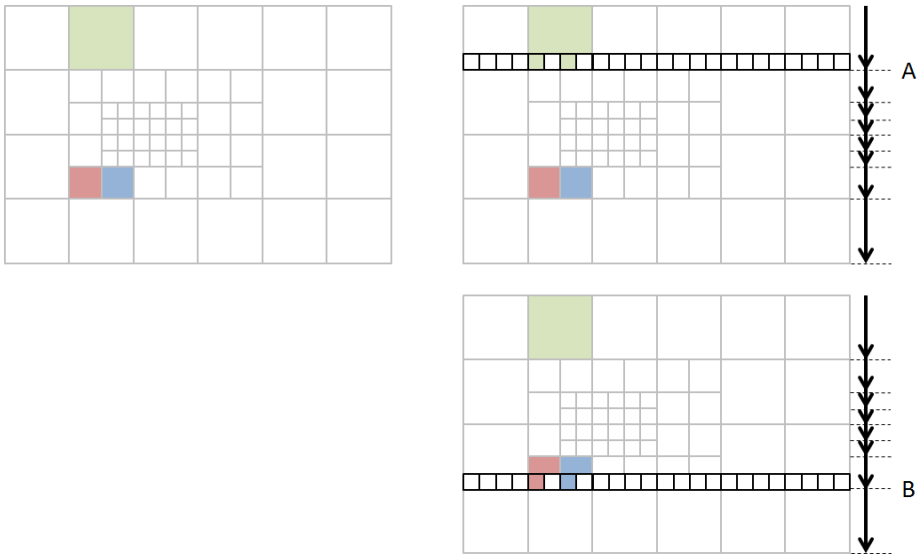


Figura 4-28: Proceso de almacenamiento de los índices de la representación multirresolución para la estimación de los vecinos superior-inferior (ver texto)

Debido a que el esquema de codificación (ver Capítulo 3) emplea el rexel de mayor tamaño de la imagen multirresolución como unidad básica, la segunda de las ventanas sólo tendrá un tamaño igual de dicho rexel (por ejemplo, en el caso mostrado en la Figura 4.25, sería de 4 bits). El esquema se programa en la versión actual de la arquitectura en la parte software, pero su diseño permitirá su sintetizado en la parte lógica, pues las ventanas se podrían mover con la generación del grafo **`BaseRAW`**. Sólo el hecho de que en el BD3P la vecindad se haya extendido para implicar también similitud en color dificulta este sintetizado, pues la distancia en color empleada si que resulta compleja de sintetizar. En cualquier caso, queda como trabajo futuro el estudiar si esta estimación de vecinos debe llevarse o no a la parte lógica.

Una vez generada la base de la estructura piramidal, la función **`GenerarNivel`** permite crear la estructura de enlaces y grafos de dimensiones sucesivamente reducidas que forman la pirámide irregular foveal. En particular, el diagrama de flujo de la primera parte de esta función, encargada de obtener los valores de *p* y *q* para cada nodo del grafo de la jerarquía, se muestra en la Figura 4-29. En dicha figura se observa que, tras un primer paso de inicialización de los valores de estas variables para todos los nodos del grafo, se determina primero si el nodo es un mínimo local (tiene un valor de varianza *v* menor al de sus vecinos). El proceso se complica pues el nodo podría tener un vecino con su mismo valor

de varianza (esto es normal al estudiar la base, donde la varianza de todos los nodos presenta el mismo valor). En ese caso, si ese vecino tiene *p* igual a 1 (es decir, se ha marcado ya como superviviente), el nodo tendrá *p* igual a 0. En caso contrario, tendrá *p* igual a 1. Una vez estimado el valor de *p*, el flujo de programa nos lleva al cálculo del valor *q* del nodo. En este caso, aquellos nodos que han sido marcados como supervivientes, o que tienen un vecino superviviente, se marcan con *q* igual a 0. En caso contrario se marcan con *q* igual a 0. Este proceso de estimación de *p* y *q* se repite para todos los nodos del grafo. Una vez terminada esta fase, se definen los nodos supervivientes como los nodos del nuevo nivel *l*+1 y se enlazan, etiquetándolos como padres (*pad*), con los nodos del nivel que los han generado en el nivel inferior. El conjunto de enlaces intra-nivel está aún claramente incompleto.

La Figura 4-29 muestra la continuación del diagrama de flujo asociado a la función **`GenerarNivel`**. En este caso se abordan secuencialmente la definición completa de los enlaces intra-nivel, la actualización de los campos de color HSV y varianza de los nodos del nivel superior, y la definición de los enlaces inter-nivel para el nivel *l*+1. Estas tres fases se enmarcan como bloques en la figura.

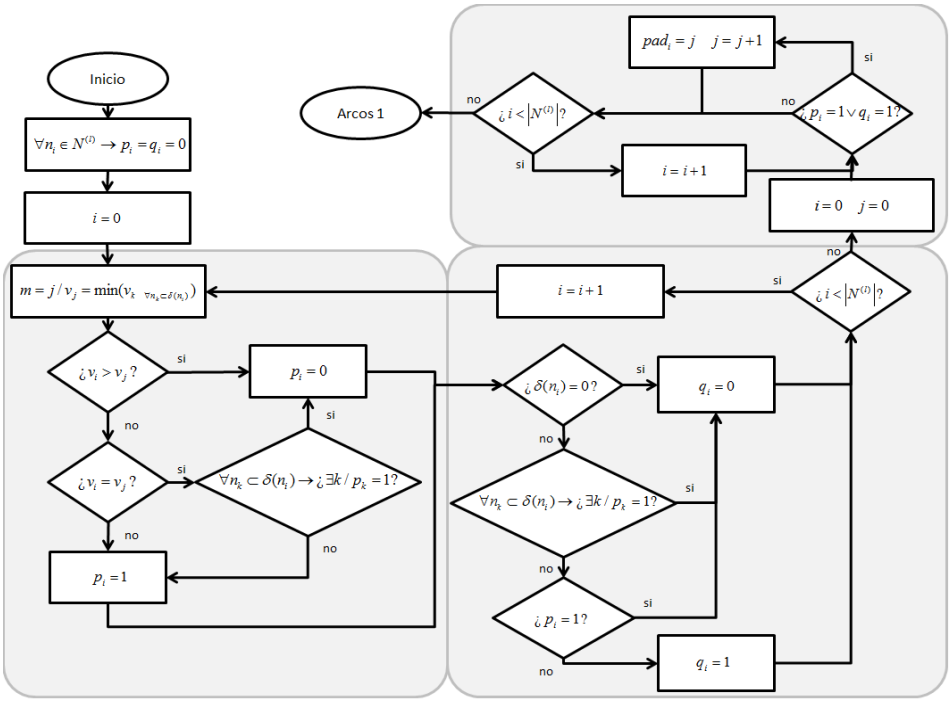


Figura 4-29: Diagrama de flujo del proceso de cálculo de los valores p y q asociados a los nodos de un grafo de la pirámide irregular foveal

La definición de los enlaces intra-nivel implica la detección, para los nodos no-supervivientes (*p* y *q* iguales a 0), del superviviente más cercano en color. El enlace se define entonces igualando el valor de padre (*pad*) del nodo al del superviviente seleccionado. El proceso se deberá repetir para todos los nodos no-supervivientes del nivel *l*. Una vez cerrado este paso se procede a actualizar los campos color HSV y varianza de los nodos que forman el nivel superior (en el diagrama no se ha detallado este proceso). Finalmente se deben establecer los enlaces que marcan las vecindades de los nodos del nivel *l*+1. Para evitar el costoso trabajo de estimar las distancias L2 y L3 (véase la descripción en el Capítulo 3), estos enlaces inter-nivel se calculan mirando, en el nivel *l*, los vecinos de cada nodo. Si el vecino de un nodo *x* en el nivel *l* tiene un padre distinto del de *x*, y estos dos padres son similares en color, se establece un enlace entre ambos padres (esto es, se convierten en vecinos) siempre que las vecindades no superen un determinado tamaño máximo (N_{max}).

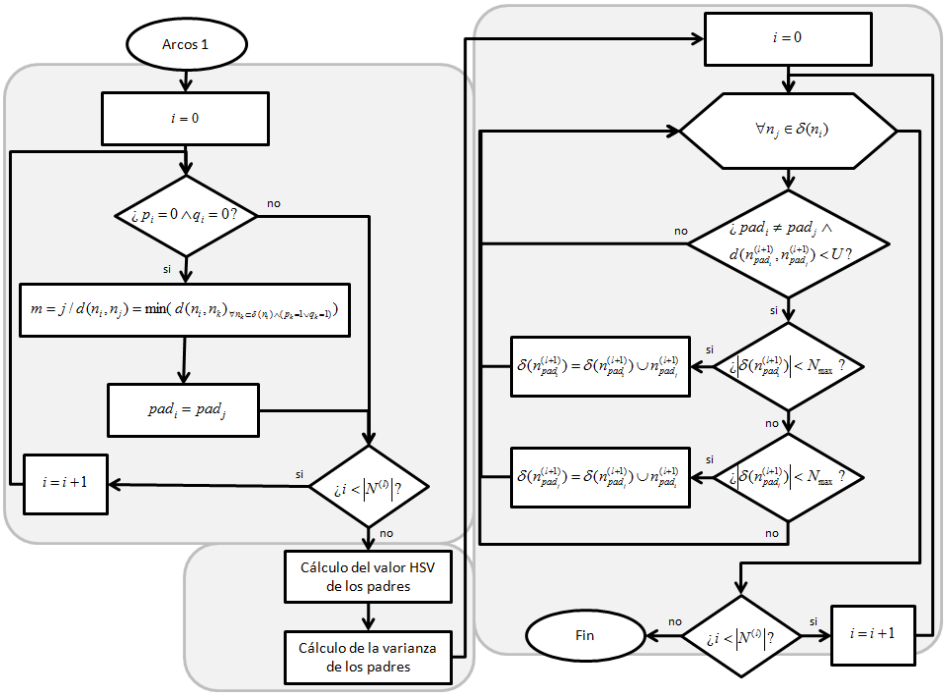


Figura 4-30: Diagrama de flujo del proceso de definición de enlaces intra- e inter-nivel y caracterización de los nodos del nivel *l*+1 en la pirámide irregular foveal

En la Figura 4-31 se muestra, finalmente, un ejemplo muy simple de funcionamiento del algoritmo de segmentación que permite reforzar algunos de los conceptos introducidos en la descripción de éste. La imagen multirresolución se codifica en un vector unidimensional, en el que dos posiciones contiguas no significan vecindad. Usando el esquema mostrado en la Figura 4.26 se seleccionan los nodos supervivientes y, usando el presentado en la Figura 4.27, se enlazan con éstos los nodos no-supervivientes. El resultado, mostrado en la imagen junto a Nivel 1, es una partición de la imagen multirresolución (los bordes se muestran con mayor grosor). Como se muestra el conjunto de supervivientes se agrupa en un nuevo vector. Se ha asumido que la distancia en color entre las tres clases no supera el umbral y un tamaño máximo de vecindad de 10. En el nivel 4 el número de clases se ha reducido a las tres presentes en la imagen. Se volvería a generar un quinto nivel, en el que el número de regiones no se modificaría, por lo que el proceso finalizaría.

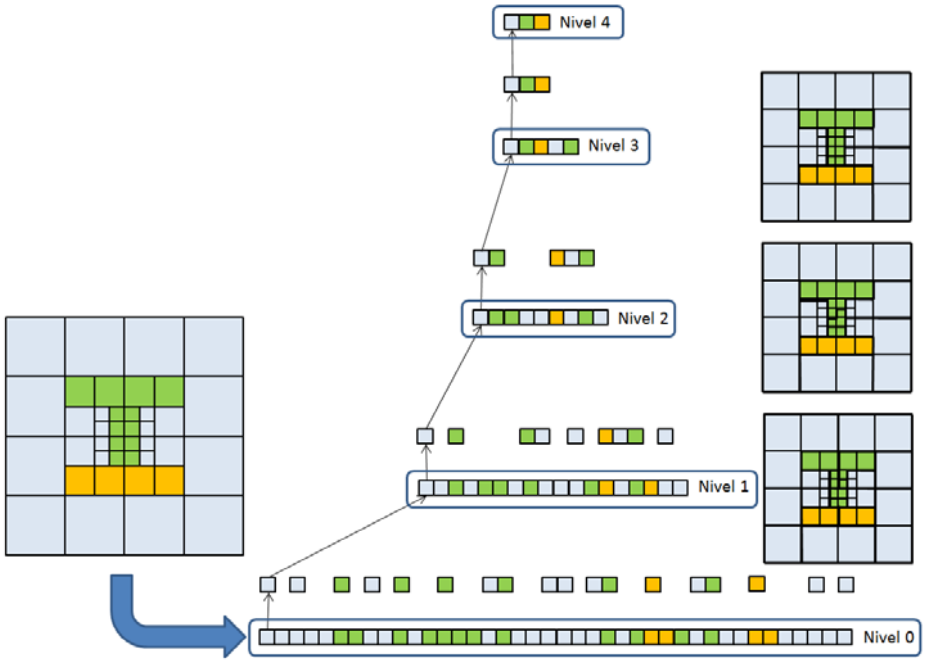


Figura 4-31: Generación de la pirámide irregular foveal para un ejemplo simple de retinotopología

4.4.2. Detalles de implementación del algoritmo de estimación del foco de atención

Respecto al método de estimación del nuevo foco de atención, basado en el concepto de mapa de saliencia, la algorítmica asociada presenta una menor complejidad pero con unos cálculos más costosos y una estructura global que es, aún si cabe, más secuencial que la asociada al segmentador. Sólo la extracción de ciertas características podría paralelizarse para ahorrar tiempo, pero al tratarse normalmente de procesos acumulativos, exige el empleo de variables pesadas en memoria. En la Figura 4-32 se muestra el diagrama de flujo de la función **`EstimarSaliencia`** y las principales variables empleadas. Se aprecian claramente los procesos acumulativos referidos, asociados al cómputo de las medias en posición *x* e *y* y el área, necesarias para definir el centro de masas, los contrastes y perímetros, los momentos, etcétera. Tanto el cálculo de los contrastes en color e intensidad y el perímetro de cada proto-objeto, como el posterior de los momentos, se lleva a cabo en un proceso acumulativo que usa el nodo como unidad. Esto permite no abordar el análisis de cada proto-objeto como un todo y simplifica globalmente el proceso (no se

recorre por ejemplo específicamente su interior). El último bucle del diagrama descrito en la figura proporciona el contraste en orientación y el valor final de saliencia de cada proto-objeto (se ha simplificado la expresión usando el vector de características f_k, que englobaría las cuatro características computadas).

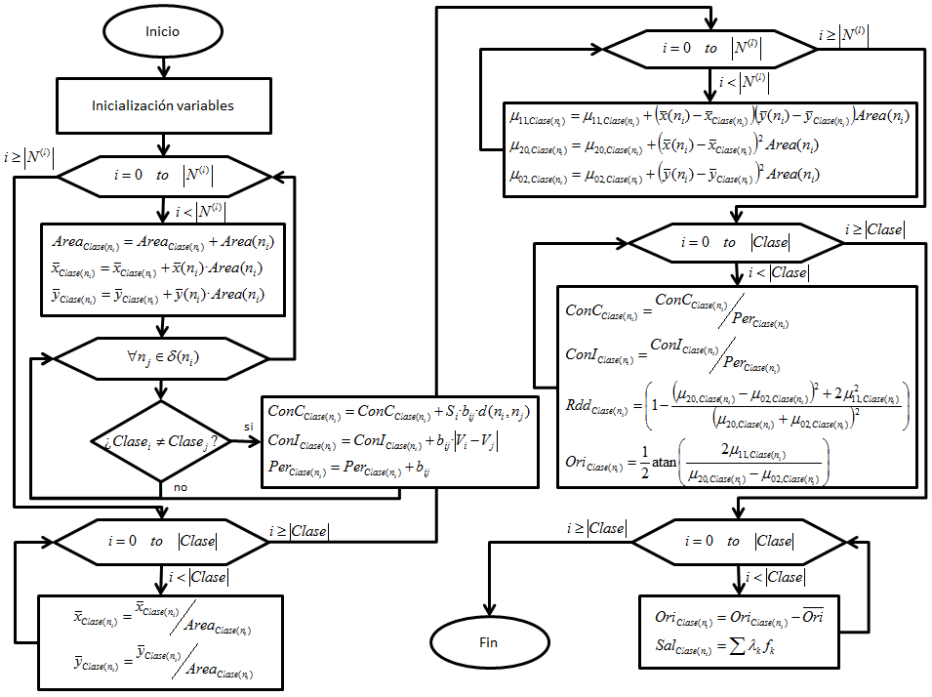


Figura 4-32: Diagrama de flujo del programa de estimación de la saliencia

Tras calcular el valor de saliencia de cada proto-objeto, la posición del foco de atención se dirige al de mayor relevancia. Al realizar ese movimiento en el campo de visión del sensor, el tamaño de la fóvea podría modificarse para adaptarse al tamaño del nuevo proto-objeto, lo cual implicaría ajustar individualmente los valores de *T*, *B*, *L* y *R*, que definen la retinotopología en una GMFD de tamaño adaptativo. Sin embargo, para mantener constante el tamaño del vector unidimensional que define **`BaseRAW`**, se ha optado por mover el centro de una fóvea de tamaño constante al centroide del proto-objeto seleccionado. Para ello, basta con mantener constantes las sumas de *T+B* y de *L+R*. Si el proto-objeto seleccionado es el asociado a la clase Clase(ni), el cálculo de estos parámetros se lleva entonces a cabo usando la expresión:

$$T = \frac{(\bar{y}_{Clase(n_i)} - medfov_y)}{\sum_i tamRing_i} \qquad L = \frac{(\bar{x}_{Clase(n_i)} - medfov_x)}{\sum_i tamRing_i}$$

$$B = numSubR_y - T \qquad R = numSubR_x - L$$

donde ($medfov_x$,$medfov_y$) es, en valores absolutos, el centro de la fóvea (por ejemplo, (92,78) si se emplea una fóvea de 184 x 156 píxeles), $tamRing_i$ x $tamRing_i$ es el tamaño de los réxeles de cada anillo i (en una retinotopología con cinco anillos de resolución, por ejemplo, el tamaño de los réxeles del quinto anillo será de 32 x 32), y $numSubR_x$ y $numSubR_y$ expresan el doble del número de subanillos en las orientaciones vertical y horizontal respectivamente (o el valor constante que suman *T+B* por un lado ($numSubR_y$) y *L+R* por el otro ($numSubR_x$)). Todos estos valores son constantes que se fijan inicialmente. Como se ha comentado, el objetivo es que el tamaño del vector asociado a la base sea de tamaño constante, lo que permite acotar los recursos en memoria que consumirá el sistema completo.

4.5. Resumen del capítulo

En este capítulo se ha presentado en detalle los fundamentos del desarrollo de cada una de las etapas que componen la cadena de preprocesado, prestando atención en la optimización de los recursos de memoria utilizados en las dos etapas que hacen uso de ellos. Aprovechar de manera óptima estos recursos disponibles, requiere de un conocimiento previo de sus características estructurales y funcionales sin el cual podemos caer en el error de proponer modelos difícilmente sintetizables, o bien sintetizables de manera muy poco óptima.

Ha quedado definido un marco de descripción de nuestros algoritmos, que encaja perfectamente en la creación de cores que se van a integrar a nivel de sistema haciendo uso del bus AXI4-STREAM, sin la necesidad de que nuestra descripción haga referencia a aspectos de bajo de nivel en la gestión de acceso a dicho bus.

La cadena de preprocesado en su conjunto permite entregar a las etapas de procesado, la imagen en el nivel de resolución que dicho procesado requiera, y como aspecto novedoso con respecto a contribuciones preliminares, con toda la información de color que nos ofrece el sensor, en espacios de color adecuados para la simple visualización o bien adecuados para el propio procesado.

También se ha detallado el desarrollo de los algoritmos de implementación software que han de ejecutarse en la parte PS del AP SoC, describiéndose el marco completo hardware/software que soporta el sistema completo, definiendo de manera clara dónde se implementa cada uno de los módulos de procesado, y cómo están comunicados entre sí, poniendo especial atención en la transferencia de datos a procesar entre la parte PL y la parte PS. La arquitectura de memoria compartida que facilita esta tarea se ha descrito describiendo su funcionamiento de manera muy esquematizada.

Capítulo 5

Pruebas y resultados experimentales

Como se esbozó en el Capítulo de Introducción, el sistema propuesto en esta Tesis Doctoral consta de tres módulos básicos: fovealizador, segmentador y estimador de saliencia. Aunque al describir el diseño de estos módulos, en los Capítulos 3 y 4 de la presente memoria, ya se ha proporcionado algún resultado, este Capítulo recoge una descripción más detallada de las pruebas de verificación de estos módulos, así como de los resultados finalmente obtenidos. Es importante notar además que, en cada Sección de este Capítulo, se irá aumentando el nivel de integración. Esto es, el algoritmo de segmentación (Sección 5.2) ya usará el fovealizador para generar el grafo usado en la base de su jerarquía, y el sistema de atención foveal (Sección 5.3) emplea los proto-objetos obtenidos por el segmentador.

5.1. El fovealizador

Este módulo agrupa toda la cadena de prepocesado que describimos en el apartado 4.2 del capitulo 4, y que está compuesta por tres etapas: la de interpolación de color, la de generación de niveles multirrresolución y la de conversión de espacio de color. En el modelado y desarrollo de cada una de las etapas se busca cumplir el doble objetivo de conseguir unos resultados de síntesis optimizados en cuanto a recursos utilizados y rendimiento, teniendo en cuenta que deben procesar datos en tiempo real que se suministran en un flujo continuo a razón de un nuevo dato cada ciclo de reloj, además de garantizar la

correcta funcionalidad. Los resultados de síntesis se han agrupado y presentado de manera progresiva para resaltar cómo las directrices de síntesis que se presentaron en las descripciones del Capítulo 4, son fundamentales para conseguir controlar el uso que se hace de los bloques de memoria interna BRAM, y para conseguir que el circuito resultante sea capaz de procesar un nuevo dato cada ciclo de reloj, o lo que es lo mismo, que el Intervalo de Iniciación de los bucles de procesamiento sea igual a 1 ciclo de reloj. La cantidad de Flip-Flop y de LUT nos darán una idea de la complejidad de la lógica necesaria. El correcto funcionamiento de las distintas etapas se ha verificado haciendo uso de un conjunto de imágenes de prueba utilizadas en su trabajo por (Malvar, 2004), tanto a nivel de descripción de alto nivel en C/C++, como a nivel de descripción RTL.

5.1.1. Interpolación de Color

La propuesta básica para este bloque contempla trabajar con el tamaño de imagen máximo que suministra nuestro sensor, sin hacer uso de directivas de síntesis, modelando un buffer de línea no optimizado, y sin un procesado particularizado de los bordes de la imagen, utilizando la interpolación básica con una ventana de dimensiones 3x3.

DSP	4	**Clock (ns)**	9,40
BRAM_18K	6	**Latencia máxima (clock cycles)**	201739291
FlipFlop	552	**Intervalo de Iniciación (clock cycles)**	201739292
LUT	895	**Interfaz**	ap_memory

Tabla 5-1: Recursos utilizados por la implementación de la propuesta básica

Cada una de las tres líneas que componen el buffer de línea requiere de 2 BRAM para poder almacenar una línea completa de la imagen original (2592 píxeles), lo que significa que uno de estos bloques está infrautilizado, pues sólo estamos usando menos de la mitad de su capacidad. Sería interesante analizar los requisitos del sistema por si fuera suficiente trabajar con una imagen de menor longitud de línea.

Los array de datos de entrada y salida de la función principal se implementan con un interfaz de acceso a memoria lo que requiere también de lógica adicional a la necesaria para el propio procesamiento. El procesamiento se realiza en esta propuesta básica de manera secuencial, lo que hace necesario

unos 2 segundos para procesar completamente una imagen. Hay por lo tanto dos aspectos que hay que controlar: el tipo de interfaz, pues el bloque tiene que procesar datos tipo stream, y hay que mejorar el rendimiento explotando el paralelismo que ofrece la FPGA.

	Propuesta básica	+ Pipeline Bucle	+ Interfaz Stream
DSP	4	5	3
BRAM_18K	6	6	6
FlipFlop	552	479	500
LUT	895	572	543
Clock (ns)	9,40	9,40	9,40
Latencia máxima (clock cycles)	201739291	5043393	5043391
Intervalo de Iniciación (clock cycles)	--	1	1
Interfaz	ap_memory	ap_memory	axis

Tabla 5-2: Comparación resultados con mejora en rendimiento y definición de interfaz

Forzando a la herramienta de síntesis a implementar el bucle principal de procesado con una arquitectura pipeline se obtiene el rendimiento máximo que nos permite procesar una nueva entrada en cada ciclo de reloj. Cuando además definimos el tipo de entrada/salida para que sea de tipo *stream*, se obtiene una simplificación en la lógica de control de dicho interfaz reflejada en una reducción de los recursos utilizados

Esta implementación representa la propuesta básica definitiva, sobre la que se ha realizado diversas variaciones buscando mejoras funcionales manteniendo el nivel de optimización de recursos.

5.1.1.1. Ajuste entre ancho de imagen y tamaño de BRAM

No siempre nuestra aplicación de segmentación requiere la máxima resolución de 5 megapíxeles que ofrece nuestro sensor, por lo que podemos reducir el ancho de la imagen original para que la anchura del buffer de línea coincida con la dimensión máxima que se puede implementar en un solo BRAM (en nuestro caso 2Kx8). Esto nos lleva a trabajar con una imagen de 3 megapíxeles (2048x1536), dimensionado que supone una reducción en la

cantidad de recursos necesarios, especialmente llamativo en el número de BRAM necesarios.

5.1.1.2. Optimización del buffer de línea

En el capítulo 4 ya se justificó que no es realmente necesario utilizar un buffer de línea completo de dimensiones N x MAXCOLS, pues el procesamiento de cada ventana no exige que la última línea del buffer esté completa, por lo que realmente sólo son necesarias N-1 líneas. De esta manera la ventana de procesado se “rellena” en sus N-1 primeras filas con valores que proceden del buffer de línea, y su fila N se rellena directamente con datos que proceden de la imagen original.

	Propuesta básica	Ajuste Imagen/BRAM	Buffer de línea optimizado
DSP	3	3	3 - 3
BRAM_18K	6	3	4 - 2
FlipFlop	500	495	520 - 514
LUT	543	538	557 - 552
Clock (ns)	9,40	9,40	9,40
Latencia máxima (clock cycles)	5043391	3149319	5043391 - 3149319
Intervalo de Iniciación (clock cycles)	1	1	1
Interfaz	axis	axis	axis

Tabla 5-3: Comparativa resultados con ajuste de dimensiones y buffer de línea optimizado

El ajuste de tamaños entre la imagen y los BRAM supone un ahorro del 50% de estos recursos, y una ligera reducción en la lógica de control de los mismos. La columna de resultados para el buffer de línea optimizado, recoge estos tanto para el caso de imagen original e imagen ajustada en tamaño. Puede llamar la atención la reducción en la latencia máxima, pero este valor es básicamente la resolución de la imagen procesada. Lo realmente importante es que el rendimiento se mantiene siempre en un ciclo de reloj por cada dato procesado.

5.1.1.3. Interpolación adaptativa

En el capítulo 4 se presentó una variante a la interpolación bilineal básica que ofrece una mejora en la calidad de la imagen reconstruida, a costa de una

estructura de datos algo más compleja pues requiere de una ventana de procesado de dimensiones 5 x 5 por lo que el buffer de línea debe tener dimensiones de 5 x MAXCOL o 4 x MAXCOL si se opta por el buffer de línea optimizado.

5.1.1.4. Procesado de los píxeles del borde de la imagen

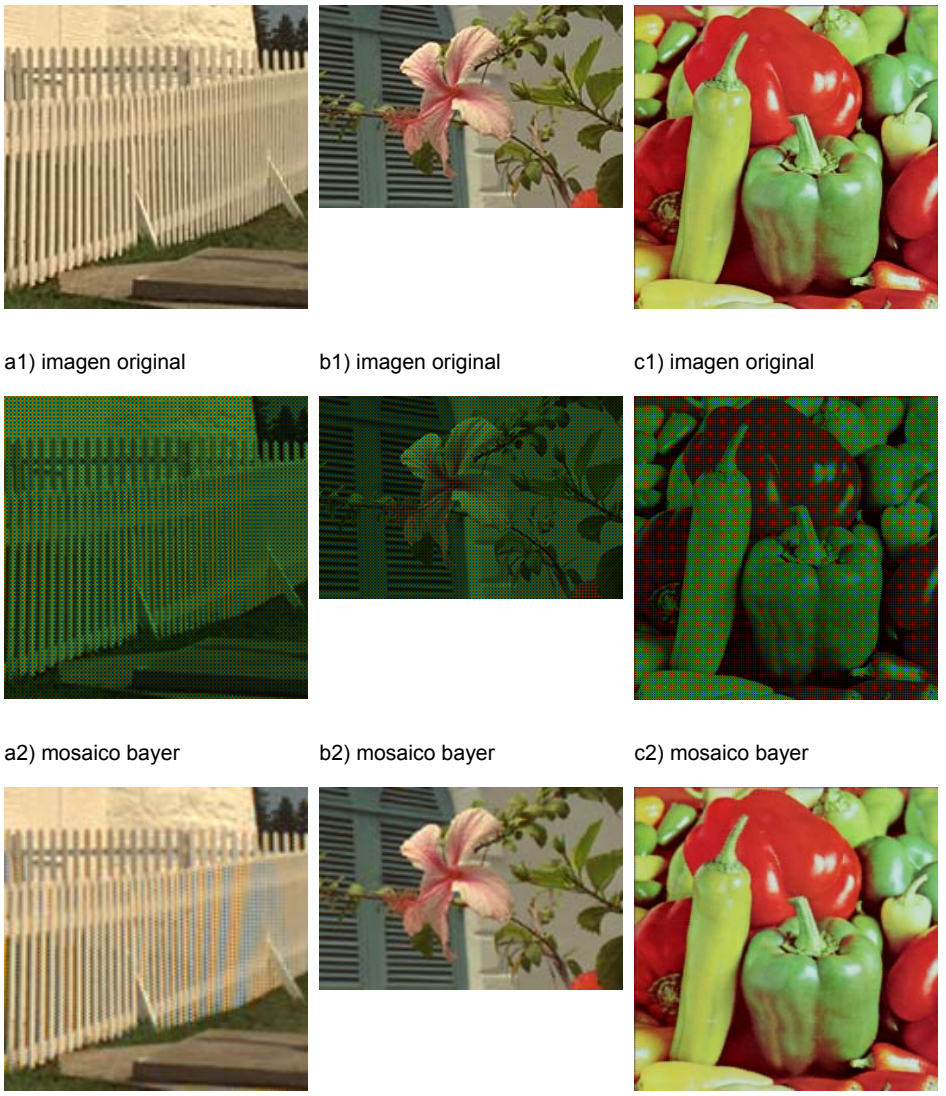
En el procesado de imágenes basado en una ventana centrada en el píxel actual que procesamos, siempre que se procesa los píxeles correspondientes al borde de la imagen, la ventana de procesado está incompleta. En la propuesta básica los datos que faltan están inicializados a 0, pero el tipo de procesado es el mismo que en los píxeles internos donde sí se dispone de la ventana completa. Esto implica la obtención de resultados erróneos para todos los píxeles del borde. La particularización del procesado en esta situación para obtener resultados correctos, implica mayor complejidad en la lógica de control y mayor número de recursos de procesado.

	Propuesta básica	Procesado de Bordes	Interpolación adaptativa
DSP	3	6	6
BRAM_18K	6	6	10
FlipFlop	500	620	638
LUT	543	580	593
Clock (ns)	9,40	9,40	9,40
Latencia máxima (clock cycles)	5043391	5043391	5043391
Intervalo de Iniciación (clock cycles)	1	1	1
Interfaz	axis	axis	axis

Tabla 5-4: Comparativa resultados con procesado particular en los bordes y con interpolación adaptativa frente a bilineal básica.

5.1.2. Verificación funcional y RTL

Las imágenes de prueba originales tienen un formato de color RGB888, y a partir de ellas se ha generado un imagen que representa el mosaico de bayer a partir del cual se realizará la reconstrucción de la información de color para cada píxel.



a1) imagen original | b1) imagen original | c1) imagen original

a2) mosaico bayer | b2) mosaico bayer | c2) mosaico bayer

a3) resultado interpolación | b3) resultado interpolación | c3) resultado interpolación

Figura 5-1: Resultados de la reconstrucción por interpolación bilineal

Los resultados muestran el correcto funcionamiento del proceso de reconstrucción, tanto en imágenes con figuras de tamaño reducido (caso de las vallas en la columna de la izquierda), como en imágenes con objetos en primer y segundo plano así como en imágenes en con objetos originalmente con bordes bien definidos. En este último caso se puede observar el efecto zipper típico en este tipo de reconstrucción de color.

5.1.3. Generación de niveles multirresolución

La propuesta básica para el módulo generador de los niveles multirresolución procesa datos de 8 bit en un flujo continuo y genera a su salida 5 niveles de resolución decreciente. Internamente está modelado como un datapath de aritmética entera para datos de anchura fija definida por el tipo estándar de C/C++ unsigned short, que es utilizado por ser el de tamaño más aproximado a los datos a procesar. El datapath no trunca los valores intermedios por lo que la anchura del mismo se va adaptando a medida que generamos los niveles de menor resolución.

	Propuesta básica	**Pipeline bucle principal**	**Interfaz Stream**
DSP	5	6	0
BRAM_18K	6	6	6
FlipFlop	513	419	409
LUT	457	419	414
Clock (ns)	7,50	9,09	5,42
Latencia máxima (clock cycles)	45353520	5038860	5038861
Intervalo de Iniciación (clock cycles)	?	1	1
Interfaz	ap_memory	ap_memory	axis

Tabla 5-5: Resultados de síntesis de la propuesta básica para uno solo de los canales de color. Los recursos totales necesarios son los equivalentes a tres canales como el ilustrado en esta tabla.

El rendimiento óptimo de una dato procesado por cada ciclo de reloj se consigue, una vez más, forzando la implementación de una estructura pipeline para el procesamiento modelado en el bucle principal. El interfaz necesario para su integración a nivel de sistema se garantiza forzando el tipo STREAM para las entradas y las salidas.

Muy interesante el informe detallado sobre el uso que se hace de los bloques de memoria y que se resume en la siguiente tabla, en la que cada una de las líneas recoge la información de relacionada con los buffer necesarios en la generación de cada nivel.

Memoria	BRAM_18K	Words	Bits
LineBuffer1	2	1296	16
LineBuffer2	1	648	16
LineBuffer3	1	324	16
LineBuffer4	1	162	16
LineBuffer5	1	81	17

Tabla 5-6: Uso de los recursos BRAM con una anchura de datapath para unsigned short

Podemos observar que la herramienta HLS no optimiza la anchura del tipo de datos utilizado, forzando la creación de estructuras de, como mínimo, la del tipo base unsigned short de 16bit. Sin embargo si la adapta para representar datos que requieran más anchura, como ocurre en el buffer utilizado en la creación del nivel 5 de resolución. Todos los bufferes encajan dentro de un solo BRAM excepto el primero que requiere de dos bloques. Esto es debido a que, como se describió en el capítulo 3, entre las configuraciones posibles de los BRAM estaban la de 2048x8 (o bien 2048x9), pero para una anchura mayor ya debemos usar 1024x16, con una profundidad insuficiente para dar cabida a las 1296 palabras necesarias.

5.1.3.1. Ajuste anchura de datapath mediante uso tipos arbitrarios

Cuando se hace necesario la optimización de la anchura de datos en el modelado del datapath, el entorno de desarrollo de la herramienta HLS nos ofrece la posibilidad de utilizar tipos predefinidos no estándar que nos permiten definir la precisión de los datos de tipo de entero, ajustándose a la anchura que realmente necesitamos en nuestro procesado, forzando al proceso de síntesis a generar la lógica necesaria con esas mismas dimensiones. Es un aspecto importante porque son muchos los registros internos que se generan cuya dimensión depende estrechamente del tipo de dato con el que se trabaja. La segunda columna de resultados de la Tabla 5-8 así lo refleja, en una reducción importante del número de Flip-Flop necesarios para implementar estos recursos. Este ajuste del tipo de datos también afecta al número de bloques BRAM necesarios para implementar el buffer de mayor tamaño que es el necesario en la generación del nivel 1. Como podemos ver en la Tabla 5-7 el análisis de los requisitos de memoria para este buffer muestra que ahora solo son necesarias palabras de 9 bit, lo que nos permite encajar esta estructura en un solo bloque BRAM de dimensión 2048x9. El resto de buffers tiene

dimensiones menores, y aunque la anchura de dato es creciente en los diferentes niveles, todas ellas encajan en bloque de BRAM de 1024x17

Memoria	BRAM_18K	Words	Bits
LineBuffer1	1	1296	9
LineBuffer2	1	648	11
LineBuffer3	1	324	13
LineBuffer4	1	162	15
LineBuffer5	1	81	17

Tabla 5-7: Uso de los recursos BRAM con una anchura de datapath ajustada mediante el uso de tipos de precisión arbitraria en función de la etapa del datapath modelada.

5.1.3.2. Optimización de uso de bloques BRAM por fusión de memorias

Otro aspecto interesante en los datos recopilados en la Tabla 5.7 es que, excepto el buffer necesario para el nivel 1, el resto requiere de un número de palabras bastante reducido, pero a pesar de ello la herramienta HLS le asigna un bloque BRAM de 1024 palabras con un desaprovechamiento importante de la capacidad de dichos bloques. El proceso de síntesis por defecto no garantiza esos niveles de optimización, que solo se pueden explorar haciendo uso de las directivas disponibles en relación con la implementación de estructuras de memoria. Entre esas directivas disponemos de la posibilidad de definir arrays de memoria grandes a partir de arrays más pequeños, por los que podemos unir, por ejemplo, los buffers de los niveles 2 y 3 por un lado, y los de los niveles 4 y 5 por otro. En cada uno de los casos, el resultado encaja en un solo bloque de 1024 palabras.

	Propuesta básica	Uso tipos precisión arbitraria	Fusión de memorias
DSP	0	0	0
BRAM_18K	6	5	3
FlipFlop	409	336	288
LUT	414	416	470
Clock (ns)	5,42	6,39	8,59
Latencia máxima (clock cycles)	5038861	5038859	5038855
Intervalo de Iniciación (clock cycles)	1	1	1
Interfaz	axis	axis	axis

Tabla 5-8: Comparativa resultados con ajuste de datapath por uso de tipos arbitrarios y optimización del uso de los bloques BRAM mediante fusión de memorias

La Tabla 5-8 recoge la comparativa que nos permite apreciar la mejora obtenida al aplicar estas dos líneas de actuación a la hora de modelar nuestro datapth y a la hora de dirigir la síntesis de memorias. Este resultado se hace más patente si se tiene en cuenta que los datos recopilados corresponden al generador de niveles multirresolución de un solo canal, y que nuestra imagen está compuesta por tres canales idénticos: uno por cada componente de color.

5.1.4. Conversión entre espacios de color

La conversión entre espacios de color propuesta en este trabajo es una transformación punto a punto que no necesita de ventana de procesado, y por lo tanto tampoco de un buffer de línea, para procesar cada uno de los valores asociados a cada píxel de entrada. Las líneas de actuación para optimizar el resultado de la síntesis se centran en las operaciones que realiza el procesado y la precisión requerida por esas operaciones. En las etapas previas, la resolución estaba bien definida por los tipos de datos que nos suministra el sensor y por la aritmética necesaria para completar su procesado, que en esos casos ha sido suficiente con aritmética entera. El procesado que requiere esta etapa básicamente consiste en transformaciones entre espacios cartesiano y cilíndrico, lo que en el desarrollo de una solución software nos lleva como elección inmediata a utilizar como tipos estándar el punto flotante, double o float en función de la precisión requerida. La herramienta HLS sintetiza correctamente las operaciones con este tipo de datos, pero hay que tener en

cuenta que la tecnología en la que vamos a implementar el resultado de la síntesis no dispone de lógica en punto flotante por lo que genera bloques funcionales a partir de los bloques DSP que básicamente realizan funciones con aritmética entera. El uso de tipos de doble precisión cuando no son necesarios tiene un impacto importantísimo en la cantidad de recursos utilizados, por lo que se hace necesario modificar nuestro modelado haciendo uso de tipos de simple precisión e incluso valorar la aritmética en punto fijo, siempre y cuando no se comprometa la funcionalidad del algoritmo.

5.1.4.1. Conversión de RGB a HSV

La propuesta básica para el procesado basada en aritmética de punto flotante de doble precisión es el punto de partida en el modelado y síntesis de esta etapa. La cantidad de recursos necesarios es importante con una temporización que no se ajusta al periodo de reloj objetivo que es de 10 ns.

	Propuesta básica (double)	float	pipeline	Int/punto fijo
LUT	5893	2148	4340	2000
FlipFlop	5365	1893	3850	1779
DSP	14	5	9	2
BRAM_18K	0	0	0	0
SRL	77	32	225	122
Clock (ns)	10,584	9,529	8,405	6,615
Latencia máxima (clock cycles)	13	75	43	26
Intervalo de Iniciación (clock cycles)	14	76	1	1
Anchura de datos del interfaz	256	128	128	32

Tabla 5-9: Comparativa resultados en función de la precisión en el tipo de datos de la aritmética en punto flotante, el uso de aritmética entera y la definición de la arquitectura pipeline

El cambio a aritmética de precisión simple supone una drástica reducción en la cantidad de recursos utilizados, tanto en unidades DSP como en el resto de lógica. La mayor simplicidad que supone reducir la precisión redunda en una temporización menos ajustada pero con una latencia que se ha quintuplicado. El intervalo de iniciación tan elevado hace inviable esta solución pues los datos a procesar llegan a un ritmo mucho mayor. Para alcanzar el objetivo de un intervalo de iniciación de 1, hay que forzar la síntesis de un arquitectura

pipeline para la unidad de procesamiento. Esto siempre supone un aumento de recursos utilizados, debido al particionado de las unidades de procesado en otras más simples añadiendo etapas de registros intermedios para la correcta sincronización de las distintas etapas de la estructura pipeline. Sin embargo es la única manera de conseguir el objeto en cuanto a rendimiento, tal y como podemos ver en la Tabla 5-9. La anchura de datos del inferfaz, en el caso de la aritmética entera, refleja la restricción a unsigned char de cada uno de los canales para “encajar” con la anchura del bus AXI-stream.

5.1.4.2. Conversión de HSV a RGB

La conversión en sentido contrario, es bastante más compleja que la que acabamos de analizar. La propuesta básica para el procesado basada en aritmética de punto flotante de doble precisión es el punto de partida en el modelado y síntesis de esta etapa. La cantidad de recursos necesarios es importante con una temporización muy ajustada al periodo de reloj objetivo que es de 10 ns.

	Propuesta básica (double)	float	pipeline	Int/punto fijo
LUT	7493	5283	6224	290
FlipFlop	6755	3808	4890	126
DSP	42	24	43	4
BRAM_18K	2	2	2	0
SRL	92	37	253	0
Clock (ns)	9,641	9,426	9,928	8,914
Latencia máxima (clock cycles)	125	76	44	4
Intervalo de Iniciación (clock cycles)	126	77	1	1
Anchura de datos del interfaz	256	128	128	32

Tabla 5-10: Comparativa resultados en función de la precisión en el tipo de datos de la aritmética en punto flotante, el uso de aritmética entera y la definición de la arquitectura pipeline

El cambio a aritmética de precisión simple supone una drástica reducción en la cantidad de recursos utilizados, tanto en unidades DSP como en el resto de lógica. La mayor simplicidad que supone reducir la precisión redunda en una temporización menos ajustada pero con una latencia que se ha quintuplicado. El intervalo de iniciación tan elevado hace inviable esta solución pues los datos

a procesar llegan a un ritmo mucho mayor. Para alcanzar el objetivo de un intervalo de iniciación de 1, hay que forzar la síntesis de un arquitectura pipeline para la unidad de procesamiento. Esto siempre supone un aumento de recursos utilizados, debido al particionado de las unidades de procesado en otras más simples añadiendo etapas de registros intermedios para la correcta sincronización de las distintas etapas del pipeline. Sin embargo es la única manera de conseguir el objeto en cuanto a rendimiento, tal y como podemos ver en la Tabla 5.10.

Los bloques de memoria BRAM que aparecen no son debidos a necesidades de bufferes temporales, pues como hemos comentado al inicio de este apartado el procesamiento es punto a punto y no requiere de este tipo de estructuras. En este caso, los bloques BRAM son empleados para implementar lógica, solución adoptada por la herramienta HLS para implementar funciones complejas mediante tablas de búsqueda.

De especial interés es la gran diferencia de recursos requeridos por las versiones que utilizan aritmética entera/punto fijo, que como sabemos ofrece una precisión más reducida que la aritmética en punto flotante, pero teniendo en cuenta la restricción de la anchura del bus de datos AXI-stream que nos obliga a restringirlos a 8bits por cada canal, las diferencias en cuanto a precisión no se hacen tan patentes.

5.2. El algoritmo de segmentación BD3P

La evaluación del algoritmo de segmentación BD3P propuesto en el Capítulo 3 de esta Tesis Doctoral se ha llevado a cabo usando la métrica de Precisión-Exhaustividad (*Precision-Recall*) y la base de datos propuesta por el grupo de Jitendra Malik en la Universidad de Berkeley (BSDB500) (Martin et al., 2001; Arbeláez et al., 2011). En el algoritmo tenemos dos parámetros que se deben seleccionar: el umbral de color empleado para determinar que un enlace intra-nivel debe establecerse; y el número máximo de vecinos por nodo. El primero de los parámetros es totalmente crítico en el sistema, y determinará realmente los resultados de calidad en su aplicación. El segundo limitará totalmente también el resultado pero sólo si se le asigna un valor reducido. En caso contrario lo normal es que pocos nodos lleguen a tener tantos vecinos. Esto además sólo ocurre en los niveles más bajos de la jerarquía, en los que otros enlaces sí establecidos terminan por resolver la falta de algún otro enlace. En los siguientes Apartados se analiza tanto la selección de parámetros como la evaluación de los resultados tras su aplicación sobre las imágenes de la BSDB500.

Por otra parte, es importante describir cómo se aplicará el algoritmo a la segmentación de una imagen estática pues, como sabemos, se trata de una propuesta foveal, en la que debemos decidir dónde fijar la fóvea y con cuántas fijaciones se analizará la imagen. Evidentemente, en un escenario real, dinámico, en el que el sistema esté funcionando en su totalidad, el esquema seguido será el esbozado en el Capítulo 3 (Figura 3.1). Pero ahora se trata de verificar el segmentador. Para ello se ha optado por segmentar la imagen fuera de línea, usando el BD3P aplicado sobre la imagen uniforme. Las regiones de segmentación obtenidas servirán de entrada al mecanismo de atención, y seleccionaremos las cinco regiones más relevantes. El número de fijaciones puede resultar pequeño para el análisis de imágenes más complejas, pero en cualquier caso entendemos que es suficiente para el análisis de una imagen natural (de 481 x 321 píxeles, que se quedarán en 480 x 320 para ser analizada en una retinotopología de 5 anillos de resolución y fóvea) que será procesada por la arquitectura integrada en la Zedboard en menos de un segundo. No lo incluimos por ello como parámetro en nuestro estudio. Una vez seleccionadas estas cinco regiones, el método aplica el proceso de segmentación centrando la fóvea sobre cada región, obteniendo de esta forma cinco imágenes de bordes, de peso mayor conforme se asocian a réxeles de menor tamaño (Figura 5.2). Estos bordes se combinan en una imagen final, en la que se suman los valores de bordes coincidentes, como ya se propone en Martín et al (2001), dando más peso a estos contornos. Podríamos haber aplicado el método conjunto de forma iterativa, mezclando pasos de segmentación y estimación de saliencia, pero eso condicionaba mucho la búsqueda de nuevas regiones a la cercanía de la primera fóvea, polarizando los resultados y obligando a aumentar el número de fijaciones para estudiar la imagen completa. En cualquier caso, además, la evaluación es del segmentador, aunque empleamos la saliencia para ubicar más de una fóvea en la imagen. En la Figura 5.3 se muestran un par de ejemplos.

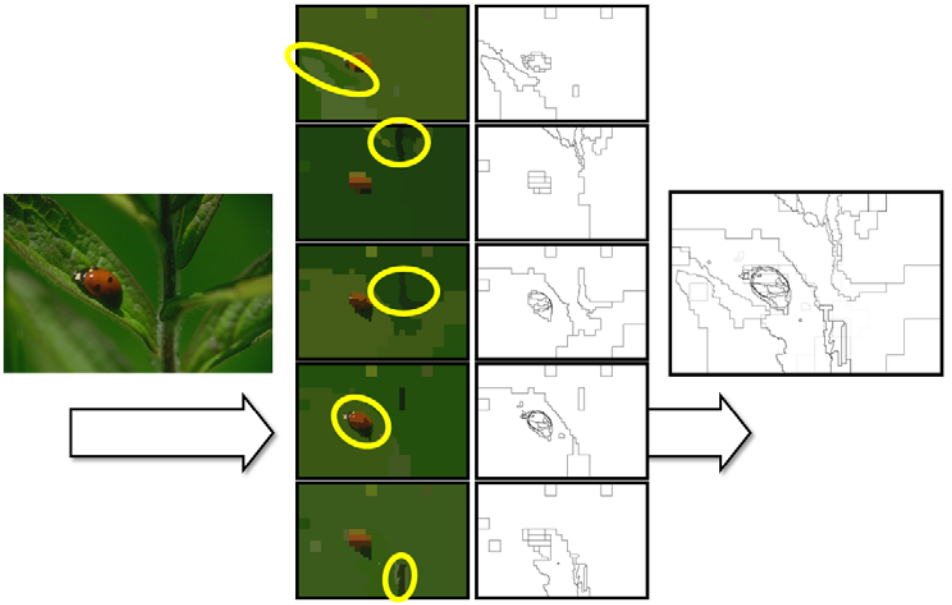


Figura 5-2: Esquema de obtención del mapa de bordes en el marco de comparación con las imágenes de la BSDB500

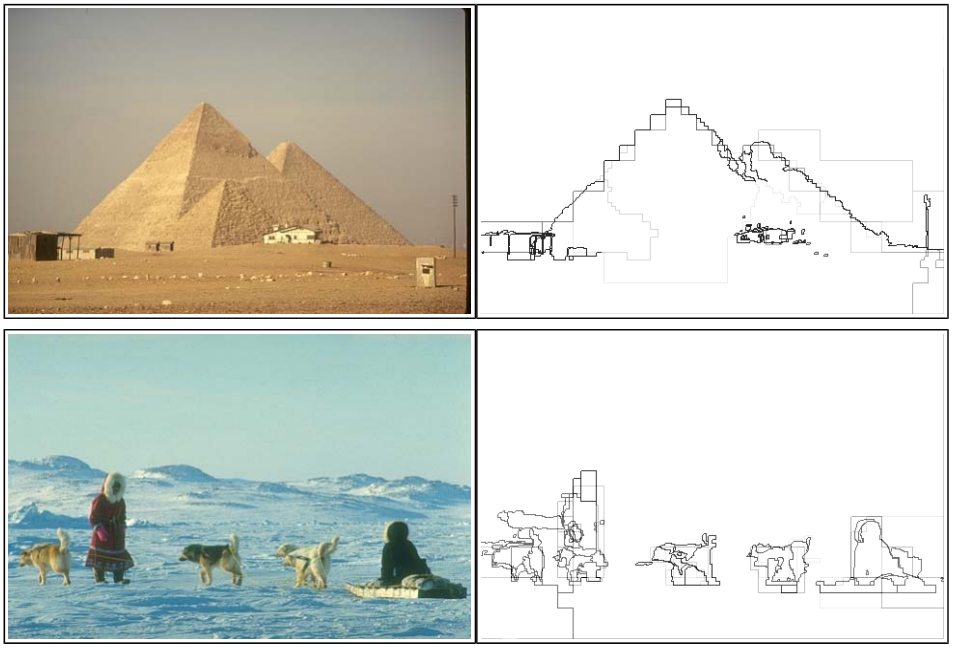


Figura 5-3: Imágenes de bordes resultantes del proceso de fovealizar con cinco fijaciones la imagen original (#161062 y #310007 de la base de datos BSDS500)

5.2.1. Selección de parámetros

La Figura 5.4 muestra los resultados obtenidos tras aplicar el método, con una distancia color de 200.0 y un valor máximo de vecinos por nodo de 10, sobre tres imágenes de la base de datos BSDS500. Son los mismos parámetros que se han empleado para tratar con las imágenes en las Figuras 5.1 y 5.2. Los parámetros empleados se han obtenido trabajando con las imágenes de aprendizaje de dicha base de datos, y son los que se han considerado como óptimos para la posterior aplicación sobre las imágenes de prueba de dicha base de datos (cuya evaluación se recoge en el Apartado 5.1.2). Las imágenes se han segmentado usando sólo los contornos de mayor peso.

En lo que respecta a la sensibilidad del método a cambios en los parámetros, el algoritmo de segmentación muestra un comportamiento robusto, que le permiten obtener con relativa facilidad una buena pareja de valores. Así, para valores de umbral que varían entre 150.0 y 250.0, y valores de vecindad máximos siempre superiores a 10, los valores de Exhaustividad son siempre superiores a 0.9. Cuando se aumenta el valor del umbral, el valor de Exhaustividad disminuye, al perderse bordes en las imágenes segmentadas de salida. Pero valores exigentes en el valor umbral también suponen aumentar el número de nodos y, por ello, la altura de la pirámide y el espacio ocupado por la estructura en memoria. Además, la Precisión disminuye, al aparecer muchos bordes, no siempre marcados como tales en las segmentaciones que se usan como referencia. De hecho, mantener valores de Precisión adecuados resulta difícil. Como se aprecia en las imágenes de resultados de la Figura 5.3, junto a los bordes reales, también marcados en las referencias o *ground-truth*, y detectados con fuerza cuando la fóvea se acerca a esas regiones, el método también devuelve muchos bordes más débiles, producto de la segmentación de esas zonas cuando están ubicadas en la región periférica del campo de visión. Esos bordes hacen descender el valor de Precisión pues, aunque la métrica de evaluación con la base BSDS500 da por buenos aquellos bordes que están cerca (en un radio de dos píxeles) de los marcados en las imágenes de referencia, usando réxeles de hasta 32 x 32 píxeles es frecuente que los bordes en la periferia no sean precisos. Es por ello que se da menos fuerza a los bordes detectados en la periferia respecto a los detectados cerca de la fóvea. La Figura 5.4 describe un ejemplo.

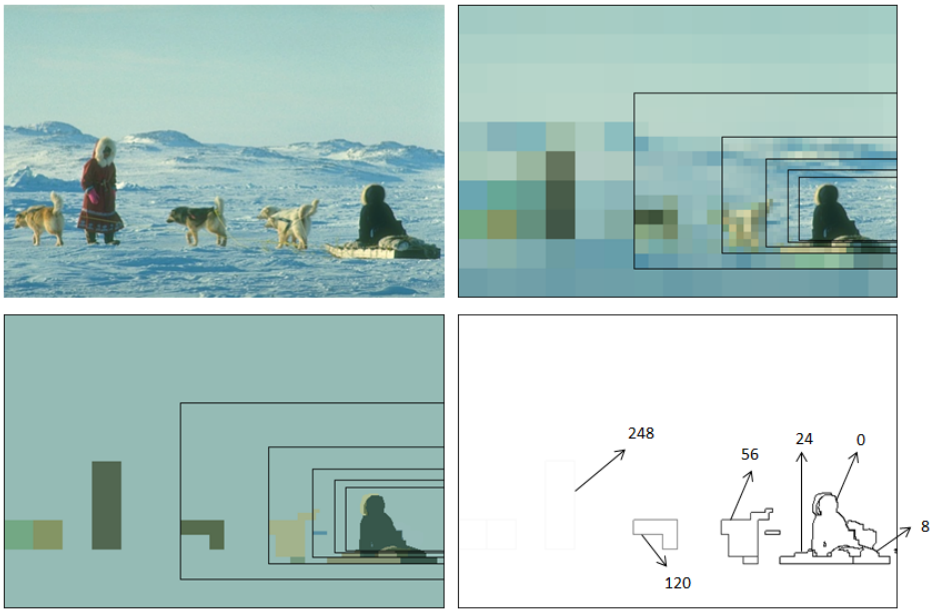


Figura 5-4: (arriba) Imagen #310007 de la base de datos BSDS500 y versión fovealizada; y (abajo) segmentación e imagen de bordes resultante

En la Figura 5.4 se muestra la imagen de bordes asociada al patrón #310007 de la BSDS500. Sobre la imagen se muestran los valores de intensidad asignados a los bordes detectados en los cinco anillos de resolución y la fóvea. Para los réxeles del anillo más periférico (32 x 32), ese valor es de 248 (en la figura no se aprecian los bordes (pero los objetos se ven en la imagen de segmentación). Para el resto de anillos los valores son 120, 56, 24 y 8. En la fóvea el valor de borde se marca en 0. Al moverse la fóvea a varias posiciones, los bordes detectados en los anillos más externo suelen ser más robustos y, finalmente, el proceso de acumulación eleva estos valores (ver Figura 5.3).

En cualquier caso, el equilibrio entre la precisión y el número de bordes detectados se alcanza con los valores de umbrales propuestos, que permiten obtener valores de calidad, evaluada como se ha comentado usando la métrica de Precisión-Exhaustividad, bastante correctos.

Por otra parte, en la gráfica de la Figura 5.5 se muestra la media de vecinos que cada nodo desprecia debido a que, pese a que el enlace no superaba el umbral de color, se alcanzaba un número de vecinos igual a 15. Se muestran sólo los niveles bajos de la pirámide, pues sólo en ellos se desprecian vecinos.

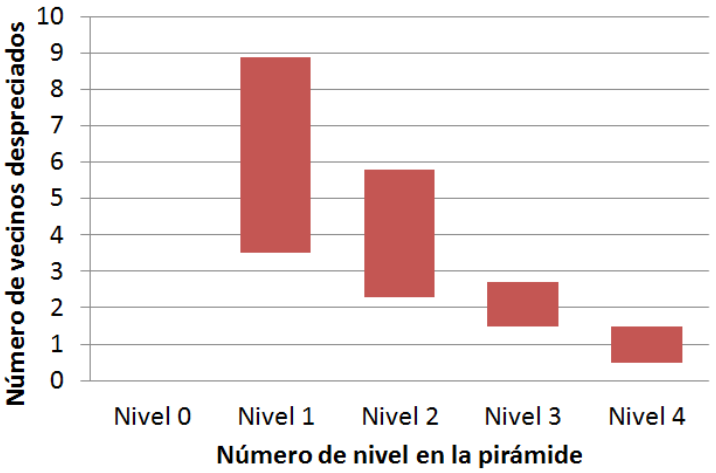


Figura 5-5: Número de vecinos despreciados por nodo y nivel en pruebas de segmentación de 10 imágenes de 480 x 320 píxeles de la base de datos BSDS500. La franja roja muestra la variación en torno a la media, obtenida usando el promedio y la varianza en la distribución.

En la Figura 5.5 se aprecia que, obviamente, en la base de la pirámide no se desprecian vecinos (la vecindad de partida impone menos de 15 vecinos por nodo). En los siguientes cuatro niveles sí que hay desprecio de vecinos que deberían ser contempladas, pero a partir del nivel cinco este valor es prácticamente cero. Estas medias coinciden con las que se han obtenido al trabajar con imágenes de otros tamaños, menores o mayores. El problema es local al nodo y no se ve afectado al aumentar el marco global de la imagen.

5.2.2. Evaluación usando la base de datos BSDB500

La Figura 5.6 muestra una primera evaluación del método usando las imágenes de prueba de la base de datos BSDB500. Aunque las pruebas realizadas sobre el conjunto de 300 imágenes de aprendizaje nos indicaron que el mejor valor de umbral de distancia era 200, se han repetido las pruebas sobre el conjunto de prueba con valores en torno a éste. En concreto, en la figura se recogen los datos obtenidos con valores umbrales de 180 y 220. Se aprecia como los valores son similares y siempre correctos, aunque los mejores se obtienen, como con el conjunto inicial de imágenes, con el valor seleccionado.

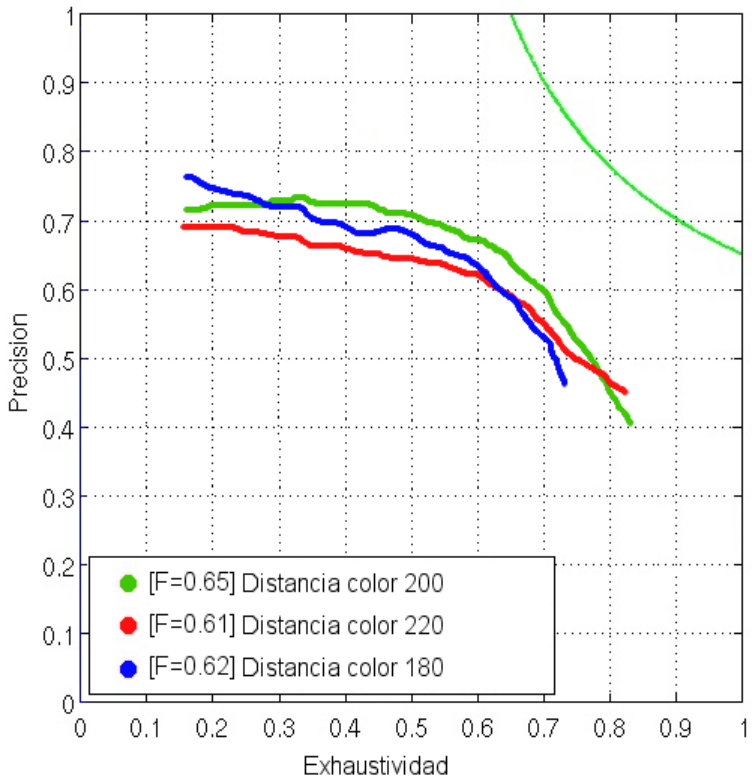


Figura 5-6: Evaluación del algoritmo de segmentación modificando el valor de la distancia color. La evaluación se ha repetido con saltos en el umbral de 10 puntos. Los mejores resultados se obtuvieron para un valor umbral de 200

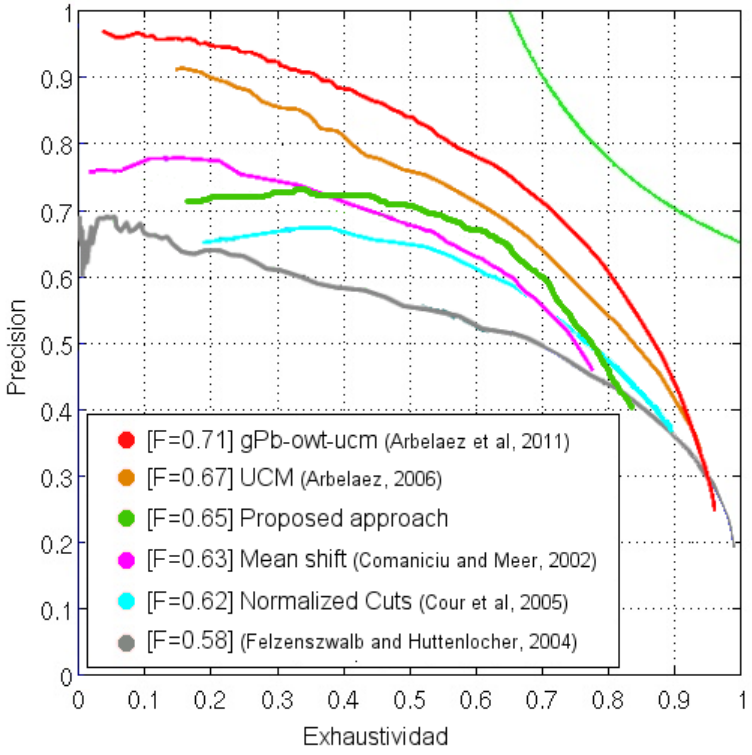


Figura 5-7: Comparación de nuestra propuesta con otras aproximaciones. Las curvas que, de otros métodos, se muestran en la figura han sido descargadas de http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/ (Arbeláez et al., 2011)

Por su parte, la Figura 5.7 permite comparar la calidad del método de segmentación implementado con las que ofrecen otras propuestas. Nuestro algoritmo de segmentación es sólo superado por el gPb-owt-ucm (Arbeláez et al., 2011) y el UCM (Arbélaez, 2006), proporcionando mejores resultados que otros métodos (Cour et al, 2005; Felzenszwalb and Huttenlocher, 2004; Comaniciu and Meer, 2002). Con respecto a los resultados que ofrecen estos métodos, se puede apreciar, en una evaluación sólo cualitativa, que la propuesta basada en grafos de Pedro Felzenszwalb y Huttenlocher (2004) y el *mean-shift* de Dorin Comaniciu y Peter Meer (2002) generan segmentaciones que capturan normalmente regiones pequeñas, de alto contraste con sus vecindades. La Figura 5.8 muestra los resultados de segmentación de ambos métodos para una imagen de la base de datos BSDS500. Usando los valores que los propios autores proponen como punto de partida, se han probado distintas combinaciones para obtener los mejores resultados en la medida F (la media armónica de los valores de Precisión y Exhaustividad). Como ocurre a veces con nuestra propuesta, es fácil que estos métodos tiendan a producir

sobre-segmentaciones de la imagen de entrada. Por el contrario, otras aproximaciones, como la de corte normalizado de Cour et al. (2005) suele caer en un excesivo agrupamiento, que combina finalmente regiones de colores distintos. El método gPb-owt-ucm, desarrollado por Pablo Arbeláez en el marco de la Universidad de Berkeley, es un método tremendamente robusto, que sólo sufre de aquellos problemas que son inherentes a un detector de bordes (variaciones intra-región fuertes pueden evitar en exceso el agrupamiento de regiones, mientras que si estas variaciones son débiles lo facilitarán, también en exceso).

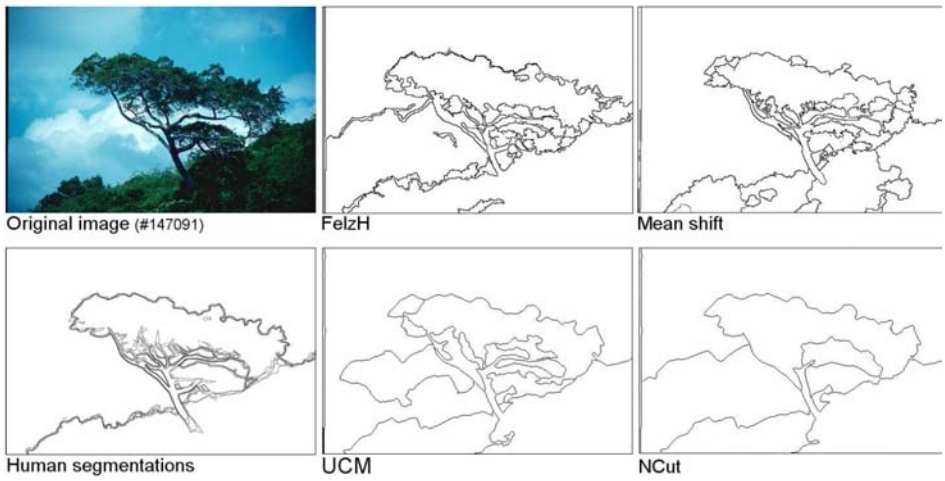


Figura 5-8: Imagen #147091 de la BSDS500 y su segmentación, tanto por personas, como por los métodos propuestos por Felzenszwalb yHuttenlocher (2004), Comaniciu y Meer (2002), Arbeláez (2006), y Cour et al. (2005). Para cada método se muestran las mejores segmentaciones de acuerdo al valor de medida F.

5.3. El mecanismo de atención foveal

Con el objetivo de evaluar las características del sistema de atención foveal propuesto en esta Tesis Doctoral, en el que incluimos el fovealizador y el segmentador como ya explicamos al principio de este Capítulo, se van a realizar tres baterías de pruebas: (a) la comparación entre usar un sensado uniforme en el espacio o un sensado variante o foveal; (b) la evaluación de la habilidad del sistema para guiar una exploración activa de la imagen; y (c) la evaluación de los modelos predictivos de atención y fijación, completando el estudio ya comenzado en el Capítulo 3 (Apartado 3.2.5), con una comparativa con otras propuestas anteriores. Cuando se emplee un método distinto al

propuesto, su evaluación se hará sobre un Intel® Core™ 2 Duo CPU T8100 2.10GHz.

Finalmente comentar que el método propuesto no dispone de un mecanismo de inhibición de retorno, que obligue a la fóvea a moverse del objeto ya enmarcado a uno nuevo. En estas pruebas esa inhibición se ha emulado anulando por software la última fóvea visitada en el mapa de saliencia, usando para ello una máscara que se almacena como variable global en el código en la parte software. En los trabajos de Antonio J. Palomino (2014) o Rebeca Marfil et al. (2014) se analiza una posible implementación e integración de esta inhibición de retorno en un sistema de atención muy similar al propuesto en esta Tesis Doctoral.

5.3.1. Sensado uniforme vs. sensado foveal

Una de las principales razones para usar una estrategia foveal es la reducción de los costes computacionales. En nuestro caso, es prácticamente imposible hacer una comparación de ambas propuestas sobre el marco del AP SoC, pues la versión uniforme no puede ser sintetizada en ningún SoC en el mercado. Pero si se aborda la comparativa de ambas propuestas en un ordenador personal (en concreto usando el anteriormente descrito), la aproximación foveal es más de diez veces más rápida que la uniforme, valor que dependerá del tamaño de la fóvea y también del tamaño de la imagen de entrada (en estas pruebas se han analizado imágenes de 1920 x 1024 píxeles). En la Zedboard, la arquitectura completa es capaz de procesar hasta 2-3 imágenes por segundo. Es importante destacar, sin embargo, que en estas pruebas en el ordenador, la obtención de la imagen foveal debe emularse por software, lo que encarece enormemente en coste y tiempo ese proceso. Los tiempos totales en el ordenador personal llegan a superar los 2 segundos por imagen.

La cuestión es entonces ¿existe un coste por ser más rápido? La Figura 5.9 muestra la comparación entre dos secuencias de fijaciones obtenidas por un modelo atencional que usa imágenes foveales (izquierda) o imágenes uniformes (derecha). No corresponden a, exactamente, la misma secuencia de vídeo, pues el método trabaja sobre los fotogramas de la secuencia que devuelve la cámara. Esto hace que, aunque el escenario sí que es el mismo, pueda haber cambios debidos a variaciones en la iluminación o a pequeños movimientos. En ambos casos, los parámetros del mecanismo de atención son los mismos, y los resultados son también muy similares. Se aprecia como, en el caso foveal, la fóvea se mueve al panel rojo en la pared (con altos contrastes respecto al fondo y de redondez), y después a la pelota roja sobre el panel verde (igualmente contrastado y redondo). El siguiente salto es a la etiqueta

amarillo-naranja del extintor. De ahí saltará a la señal de aviso de extintor, ubicada sobre el mismo. En el caso uniforme, todas las pelotas rojas sobre la pared blanca son visibles (no es el caso del caso foveal, pues las bolas rojas lejanas a la primera fovealización -el panel rojo-, se funden con el fondo al ubicarse en el último anillo de resolución), y la fóvea se mueve de una a otra. En la última imagen, sin embargo, se aprecia como el siguiente salto sería a la señal de extintor.

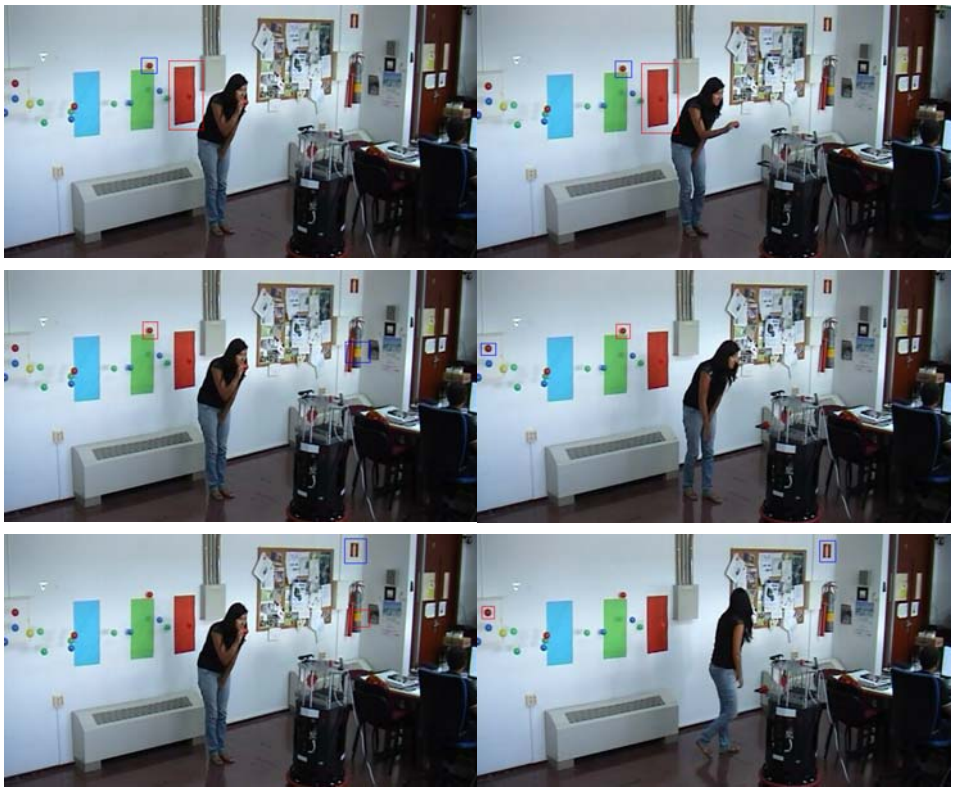


Figura 5-9: Exploración activa de una secuencia de vídeo (izquierda) imágenes foveales, y (derecha) imágenes uniformes. En ambos casos los parámetros usados en las etapas del sistema son los mismos. En rojo se muestra la posición de la fóvea, mientras que en azul la posición a la que saltará en el siguiente fotograma.

Por el contrario, la desventaja de ser lento en un marco real y dinámico son claras. En la Figura 5.9 se aprecia como en la columna izquierda todas las imágenes parecen corresponder al mismo fotograma: el sistema explora la escena a una velocidad suficiente para extraer información sin que la persona prácticamente se mueva. En la columna derecha, sin embargo, se puede apreciar claramente el movimiento de la persona. La exploración de la escena es lenta, al menos más lenta que el propio dinamismo de la misma.

5.3.2. Exploración activa usando la aproximación atentiva foveal

Una de las cuestiones que debe destacarse es que la naturaleza foveal del método hace que, de la exploración de una escena, incluso estática, el método no genera un único y fijo mapa de saliencia, sino una secuencia de mapas de saliencia. Por ello, existe un flujo de trabajo iterativo cuyos pasos implican (a) mover la fóvea a la nueva posición, (b) obtener un mapa de saliencia nuevo, y (c) determinar la posición nueva de la fóvea en función de ese mapa. La aproximación foveal debe por tanto entenderse por ello, como propia del marco de trabajo con secuencias de vídeo, esto es, en escenarios en los que la información visual cambia constantemente debido a movimientos del propio sensor (ego-céntricos) o al dinamismo de la escena (Borji e Itti, 2013b). Cuando usamos nuestra propuesta para explorar una escena estática el resultado será el mismo: es necesario más de una iteración para explorar la escena (como ya expusimos en la introducción al Apartado 5.2). Con objeto de trabajar sobre bases de datos conocidas, evaluaremos en este Apartado esta opción. Así, la Figura 5.10 muestra trayectorias de exploración (*scan-paths*) para tres imágenes distintas de la ToolBox Saliency (http://www.saliencytoolbox.net/). La columna a la izquierda muestra los resultados obtenidos usando el método propuesto por Walther y Koch (2006), mientras que la derecha presenta los proto-objetos proporcionados por nuestra propuesta. El orden seguido por las fijaciones se ha dibujado sobre las imágenes. Cada fijación se corresponde con una iteración y con el análisis de los datos de una determinada región, que será enmarcada por la fóvea. Esta exploración es un proceso activo, que es completado en un número finito de iteraciones, en las que todas las regiones relevantes de la escena han sido localizadas en la fóvea. En parte, este movimiento de la fóvea es fruto de la existencia de un mecanismo de inhibición de retorno (emulado en este caso, como ya se ha comentado). Pero también en parte, este comportamiento es fruto de la naturaleza foveal de la propuesta que, como se ha comentado anteriormente, genera segmentaciones distintas en cada iteración.

Esta característica se documenta de nuevo en la Figura 5.11. Similar a las Figuras 5.9 y 5.10, esta figura muestra, desde la imagen en la esquina superior-izquierda a la situada en la esquina inferior-derecha, una secuencia de fijaciones. La primera fóvea se localiza sobre la cara del hombre, después se mueve a la mano del hombre, la cara de la mujer, etcétera. En ocasiones la trayectoria de fijaciones no sigue el camino deseado: de la mano de la mujer se mueve a la flor junto a la cabeza del hombre. Pero hay que entender que estamos en un marco realmente activo, y el sistema volverá a regiones que

consideramos, subjetivamente, como relevantes en las siguientes iteraciones. Los resultados son similares a los proporcionados por Walther y Koch (2006) (Figura 5.12).

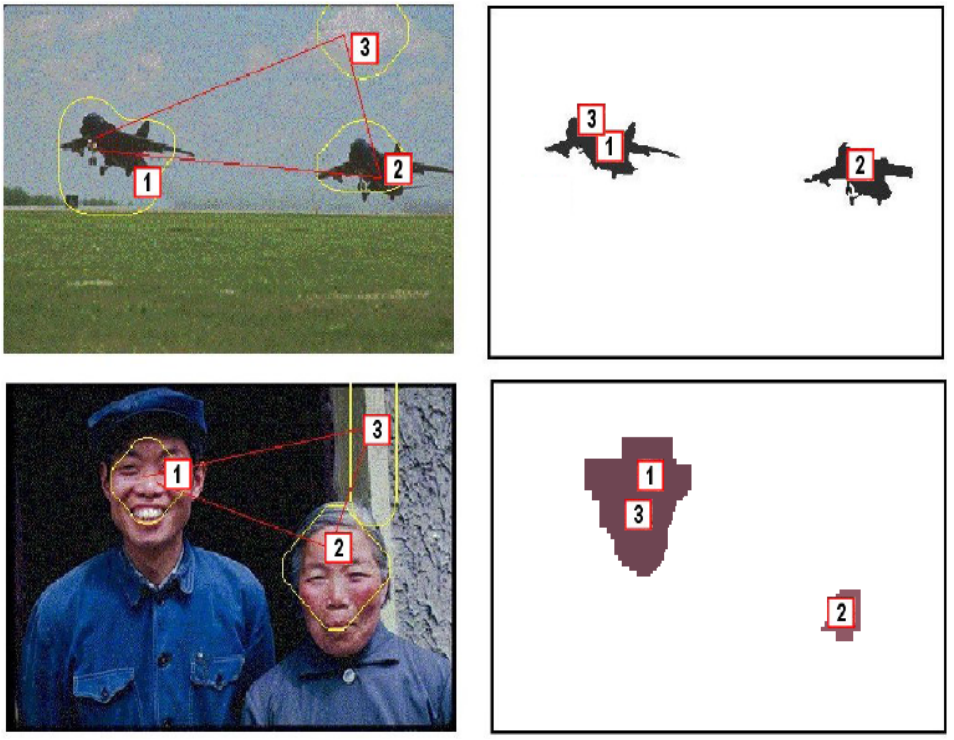


Figura 5-10: Resultados de trayectorias de fijaciones para dos imágenes de la Toolbox Saliency: (izquierda) resultados obtenidos usando el método de Walther y Koch (2006), y (derecha) proto-objetos obtenidos usando el método propuesto. El orden seguido por las fijaciones se muestra sobre las imágenes.

La efectividad de nuestra propuesta se ha verificado finalmente con experimentos que se basan en datos de seguimiento de la mirada en personas. Para compararnos con ellos se usará, como patrón de referencia, los datos de la base de datos JUDD, disponibles y abiertos (Judd et al., 2009b). Esta base de datos cuenta con grabaciones de seguimiento ocular de personas en un escenario de exploración sin tarea específica (1003 imágenes con trayectorias atencionales de 15 personas). Para compararnos, nuestras secuencias se han convertido también en un vector de posiciones en la imagen, usando para ello como posición la del centroide de la región a atender. Usando el índice de similitud propuesto por Liu et al. (2013) es posible comparar el patrón de referencia y la secuencia obtenida por nuestro método. Esta métrica emplea un parámetro de salto (gap) que penaliza cuando es necesario insertar o eliminar

una posición en la trayectoria durante el proceso de emparejamiento de secuencias. Siguiendo las indicaciones dadas por los desarrolladores de esta métrica, este salto se ha fijado en -0,5 en nuestras pruebas.

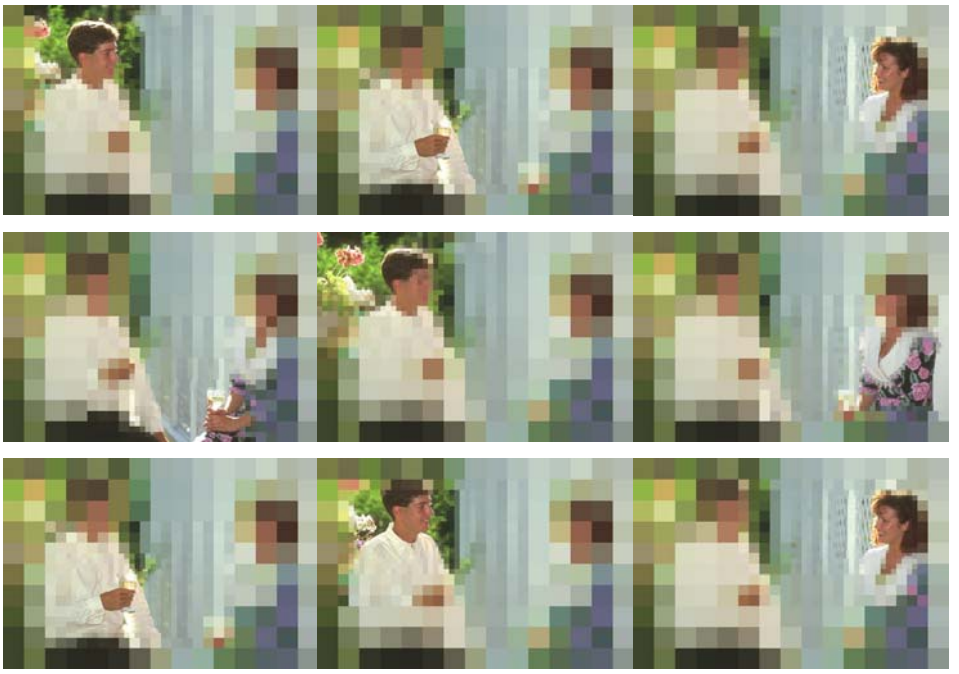


Figura 5-11: Exploración activa de la imagen #157055 de la BSDS500: desde la esquina superior-izquierda a la inferior-derecha, la figura muestra una secuencia de fijaciones.

Finalmente se compara la secuencia de posiciones que obtenemos con los 15 patrones de referencia (uno por persona), y se hace posteriormente un promediado. El resultado es mejor conforme más alto. Así, el valor obtenido es de 0,95, mejor al obtenido por los métodos de Itti et al. (1998) o Walther y Koch (2006) (ambos por debajo de 0,9). Por el contrario, el valor está por debajo del obtenido por Liu et al. (2013), que es cercano a 1,15. Sin embargo, es importante resaltar que este método usa características de bajo nivel pero también información de más alto nivel (contexto semántico), que no son tenidas en cuenta en nuestro caso.

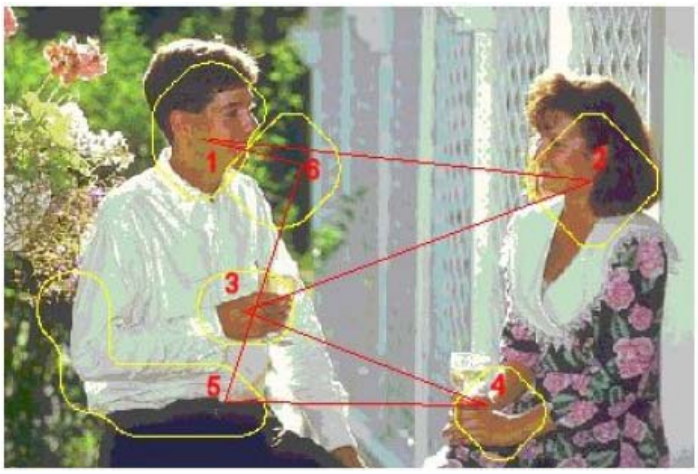


Figura 5-12: Trayectorias de fijaciones en la imagen #157055 de la BSDS500 usando el método propuesto por Walther y Koch (2006).

5.3.3. Experimentos con modelos predictivos de atención y fijación

Como referimos en el Apartado 3.2.5 del Capítulo 3, el método ha sido evaluado usando la base de datos Toronto (Bruce y Tsotsos, 2009). En dicho Apartado sólo proporcionamos las imágenes de mapas de densidad de fijaciones, obtenidos de nuestros mapas de saliencia, que podían ser comparados cualitativamente con los obtenidos a partir de datos de seguimientos de mirada en personas. Para cerrar cuantitativamente dicho estudio se evaluará el área bajo la curva Característica Operativa del Receptor (ROC, *receiver operating characteristic*), obtenida ésta como una medida del ratio de verdaderos positivos frente al de falsos positivos cuando se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo). Para ello, cada mapa de saliencia que proporciona nuestro sistema es umbralizado y considerado entonces como un clasificador binario que separa muestras positivas (fijaciones de todas las personas en esa imagen) de muestras negativas (fijaciones de todas las personas en todas las otras imágenes en la base de datos). Este proceso evita el efecto de sesgo atencional al centro de la imagen, tan documentado en atención visual (Borji e Itti, 2013b). Moviendo el valor empleado para umbralizar los mapas de saliencia es posible estimar la curva ROC y calcular después el área bajo ésta. Este área es el indicador o métrica empleado para comparar cuantitativamente cómo se asemejan el mapa de saliencia obtenido por nuestro métodos y el

mapa de densidad de fijaciones obtenido de los datos de personas. Un valor superior a 0,5 indica una correlación positiva, siendo el límite superior fijado por lo bien que las fijaciones de una persona predicen las del resto. Este límite está fijado en 0,878 (Borji e Itti, 2013b). En nuestro caso el valor obtenido ha sido 0,669, lo que lo sitúa en el quinto lugar de los 28 modelos evaluados por Borji e Itti (2013a) (Figura 5.13).

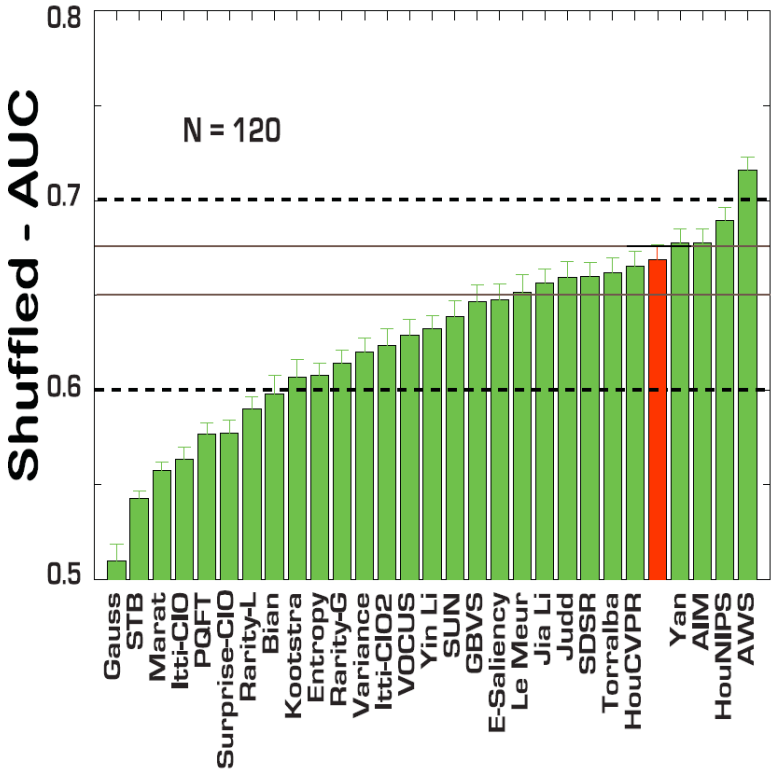


Figura 5-13: Evaluación de métodos usando la base de datos Toronto y el método del área bajo la curva ROC con eliminación del sesgo central (shuffled AUC). El método propuesto se marca en rojo entre el resto de aproximaciones (adaptación del original en Borji et al., 2013a)

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Las bases de la visión activa se asientan sobre la existencia de un mecanismo capaz de detectar cuáles son las regiones de interés en la escena, de forma que posteriormente sea posible dirigir el foco de atención sobre ellas. El objetivo es que, con una capacidad de computo limitada, se pueda procesar la información de una escena consiguiendo, como resultado, la sensación de percepción total y en tiempo real de la escena completa que comentábamos en los primeros párrafos del Capítulo 2 de la presente memoria. Sin embargo, en esta Tesis, han sido dos las líneas de trabajo que, complementarias a la puramente algorítmica del diseño del mecanismo de control de la mirada, han sido estudiadas en profundidad: el diseño de toda la arquitectura necesaria (captura y preprocesado de la imagen, estimación del mapa de saliencia, movimiento de la fóvea) y su implementación en un dispositivo dedicado, un AP SoC de altas prestaciones; y el desarrollo concreto de un hilo de trabajo en hardware capaz de generar las imágenes multirresolución color que serán la base del sistema perceptivo completo. Pese a que el mecanismo de estimación de saliencia desarrollado en esta Tesis se imbrica con un segmentador novedoso, esta parte del sistema es actualmente la más débil. Los resultados que ofrecen, tanto el segmentador como el propio mecanismo de cálculo de saliencia, son muy positivos. Pero queda trabajo por recorrer, tanto en lo que respecta al segmentador, computacionalmente optimizado para su futura

implementación en hardware, como en el propio mecanismo de atención, que no incluye elementos básicos ya referidos como la inhibición de retorno.

En lo que se refiere a la generación de imágenes de resolución espacial variable a partir de sensores de resolución uniforme, podemos afirmar que se han actualizado y mejorado en gran medida las contribuciones que datan de hace ya una década. Esta mejora se asienta en la evolución tecnológica tanto en dispositivos hardware como en herramientas software de desarrollo. Hemos podido evolucionar así a sensores de varios megapíxeles que nos ofrecen simultáneamente un campo de visión amplio con una elevada resolución. También se ha podido introducir, como innovación con respecto a aquellos primeros trabajos, el uso del color como una fuente de información relevante para cierto tipos de procesado de mayor nivel de abstracción. La plataforma propuesta e implementada en la cadena de preprocesado, define un marco de trabajo que permite disponer de una secuencia de imágenes con la resolución deseada, con la información de color requerida en cada píxel, desde los formatos estándar que ofrece el sensor hasta la información tipo RAW con la máxima resolución posible (hasta 10bits por cada canal de color).

Dentro de esta cadena de preprocesado se ha incluido una etapa que permite hacer “el revelado digital” de la imagen RAW que captura el sensor, de manera que es fácil poder cambiar el algoritmo de reconstrucción o interpolación, para obtener los resultados deseados, ya sea en calidad de imagen de color, ya sea en optimización de recursos utilizados. Incluso con los algoritmos más complejos que se han probado, se consigue entregar la imagen procesada con una latencia de tan solo unas pocas líneas de imagen, o lo que es mismo, antes de que el sensor externo empiece a captura el nuevo fotograma, nuestro interpolador de color ya ha terminado de generar la imagen reconstruida: la latencia a nivel de fotograma es, por tanto, nula.

Sin duda en este canal de preprocesado, la contribución más interesante es el módulo que genera la imagen multiresolución. Si bien los nuevos sensores nos ofrecen resoluciones de megapixeles, esta cantidad de información es excesiva para un procesado fluido que necesite tratar varios fotogramas por segundo. El módulo rexelizador, que así lo hemos bautizado, permite definir diversas retinotopologías en función de la posición donde queramos ubicar la fóvea, de la dimensión de la propia fóvea así como de su relación de aspecto. Aunque internamente el módulo está continuamente generando una pirámide regular completa con los distintos niveles de resolución decreciente, a la salida se entrega estrictamente una verdadera imagen multirresolución, con su fóvea y los distintos anillos configurados a requerimiento de la aplicación que los va a utilizar, en este caso el algoritmo de segmentación que constituye la primera

etapa del mecanismo de control de la mirada. Al igual que en el primer módulo, todo este procesamiento se realiza con latencia nula, pues en el momento en que se está recibiendo el último píxel del fotograma actual se está generando el último réxel de la imagen foveal de salida. Esta característica de latencia nula puede ser crucial en sistemas que necesiten de respuestas en tiempo real.

En el canal de preprocesado del video de entrada, la última etapa la constituye el conversor de espacio de color, necesario como ya se ha comentado, por la idoneidad de unos espacios frente a otros en función de la tarea a realizar. La conversión realizada es una transformación punto a punto, por lo que la latencia de procesado de una imagen completa es casi nula. En su desarrollo se ha explorado la posibilidad de usar aritmética en punto flotante, a sabiendas de que el dispositivo programable no dispone de unidades de procesamiento de este tipo. Aún así, se puede implementar sin problema, manteniendo la precisión que ofrece este tipo de aritmética, pero a costa de unos requerimientos hardware importantes. En esta fase de desarrollo se ha tenido que tomar decisiones de compromiso entre precisión y optimización de recursos, no sólo en el núcleo de las unidades de procesado hardware de los conversores, sino sobre todo a la hora de delimitar la anchura de los buses que han de utilizarse para permitir la buena integración con el resto del sistema HW/SW. Trabajar con punto flotante y mantener su precisión en los buses de entrada/salida exige anchuras de bus de 128bit. Se ha optado por definir una anchura de bus de 32bit, que encaja de manera óptima con la infraestructura del bus AXI que ofrece la plataforma AP SoC, a costa de perder cierta precisión.

El flujo de datos que almacenan la imagen multirresolución llegan, por tanto, desde el sensor al segmentador con una latencia prácticamente nula. Organizada en un vector unidimensional, esta imagen se trata como un grafo, en el que los enlaces entre nodos deben ser almacenadas pues la vecindad puramente espacial, que si podemos emplear en la imagen, no es válida ahora. Para ello, el segmentador debe primero calcular las vecindades en la base y, después, generar iterativamente los distintos niveles que constituyen la jerarquía en la que se ordenará el contenido de la imagen. Todo este proceso ha sido optimizado pensando en su futura implementación en hardware, buscando reducir el tiempo de procesado. Sin embargo, y sólo por no prolongar el cierre del presente trabajo, finalmente el segmentador se ha implementado en la parte software del AP SoC. El esfuerzo por optimizar su funcionamiento le permite, aún no corriendo en el hardware, ser eficiente y procesar varios fotogramas por segundo para una imagen de entrada de cerca de 40K nodos. Es importante destacar que, pese a que no lo consideramos un trabajo cerrado, los tiempos de procesado del segmentador propuesto son, en su versión actual

y en el marco general de las pirámides irregulares, mucho menores que los que conocemos, bien por conversaciones como por trabajos conjuntos, de otras propuestas en grupos como el PRIP de la Universidad Técnica de Viena (Austria) o el GreyC del ENSICAEN (Francia). El mecanismo de cómputo del mapa de saliencia se asienta, por su parte, en un conjunto típico de características cuyo sintetizado en hardware no ha sido abordado en profundidad. Es importante destacar que permite obtener resultados coherentes con los que emanan de los sistemas visuales humanos, pero a costa, en la versión actual de la arquitectura, de un tiempo relativamente alto de procesado si se compara con el resto de elementos integrados en el lazo perceptivo.

6.2. Trabajo futuro

La presente Tesis Doctoral supone, para el grupo de investigación en la que se encuadra, un hito importante, enmarcado en la continuación de la línea de investigación y desarrollo abierta ya a principios de la década de los 90s, pero, por otra parte, abriendo nuevas vías de trabajo al suponer la primera arquitectura de alto grado de complejidad que implementamos sobre un AP SoC de gama alta, como es la Zynq-7020 empotrada en la Zedboard. Esta arquitectura responde al paradigma de las cámaras inteligentes, dispositivos capaces de procesar los datos de bajo nivel para delinear descriptores de mayor nivel de abstracción que apoyen la toma de decisiones, cuestión que, como es el caso en el presente trabajo, se implementa en parte en la propia cámara.

Las líneas de investigación que continuarán el trabajo iniciado con esta Tesis supondrán

- Integración en un sistema visual activo

  La arquitectura propuesta se diseña pensando en su integración en un sistema activo, que integre un par estéreo y que incluya los elementos que ya se han reseñado faltan en su implementación actual: un mecanismo de inhibición de retorno y un módulo completo de motorización. Al contar con un par de cámaras será posible obtener la información de profundidad, descriptor básico en casi cualquier mecanismo de atención. La representación de los datos percibidos en un grafo supone un reto para la estimación de esta medida, normalmente sujeta al cumplimiento de restricciones epipolares que deberán generalizarse para que sean también tenidas en cuenta en este nuevo marco de trabajo. Queda igualmente como trabajo futuro el

diseño e implementación de fuentes de información de bajo nivel que, asociadas a la descripción del movimiento en la escena, sean susceptibles de ser integradas como generadoras de nuevos mapas de evidencias en el mecanismo de atención. La evaluación de estos descriptores en una cabeza robótica móvil supone un reto interesante, que requerirá el diseño de nuevos bloques en las partes lógicas y software del AP SoC.

Por otra parte, el motorizado de todo el sistema implicará desarrollar los componentes que integren el movimiento de la fóvea dentro del campo capturado por la cámara con el de todo el sistema, cuando el movimiento de la región a encuadrar en la fóvea lo obligue. Será el momento también de evaluar la referida ventaja de contar con una fóvea desplazable.

- Modificación de las estructuras que representan la información

  La representación de la información en un grafo supone, para las pirámides irregulares, el conseguir el grado de flexibilidad necesario para adaptar su propia estructura interna tridimensional a la distribución de datos en la imagen de entrada.

  El objetivo sería modificar la estrategia de representación de la información, sustituyendo el grafo totalmente flexible empleado en las pirámides irregulares, en el que el número de nodos y arcos es totalmente desconocido a priori, por una representación más rígida en su estructura, pero en la que la flexibilidad venga ahora de la mano de la información que se transmite por cada arco. En cierta forma, se trataría de retomar la idea de las pirámides regulares probabilísticas (Prewer y Kitchen, 2002), en las que, por ejemplo, el valor de un nodo padre no es una mera ponderación del valor de sus hijos, sino que los pesos dados a los enlaces padre-hijo permite también modular la influencia de cada nodo hijo en el valor del padre. Excesivamente costosas para su implementación software, estas estructuras podrían rediseñarse e implementarse en las nuevas plataformas AP SoC. Podríamos de esta forma montar toda la estructura de nodos y arcos desde un inicio, añadiendo un atributo o peso a los arcos. En función del contenido de la imagen estos pesos se modifican, modulando el paso de información entre nodos. Habría además que modificar el enlazado intra-nivel que, en las aproximaciones regulares probabilísticas, sigue encorsetado por las relaciones de vecindad típicas en la imagen, V4 o V8. La idea sería que, al igual que los

enlaces inter-nivel pueden abrirse o cerrarse en función de un valor de peso, la vecindad entre nodos del mismo nivel sea mayor (en la base se mantendría V4 o V8, pero dicho valor crecería conforme se subiera en la jerarquía) y también ponderada por un peso. De esta forma la flexibilidad del grafo se mantiene pues, en cierta forma, los pesos de los arcos intra-nivel permitirían trazar un grafo interno a cada nivel de la jerarquía. Solo se fijaría a priori el número de nodos y, con ello, el factor de reducción entre niveles.

- Integración en sistemas reales

  A más largo plazo, estos sistemas se integrarán en plataformas robóticas destinadas a la interacción social en entornos compartidos con personas o en drones. Esta integración dotará de pleno sentido la propuesta desarrollada en esta Tesis, al aparecer aplicaciones y tareas específicas que puedan fijar los pesos del proceso de cómputo de la saliencia, y potenciar el desarrollo e integración de la rama top-down, ahora mismo ausente, en el mecanismo de atención.

  Para que esta integración se convierta en una realidad el sistema deberá ser realmente práctico, para lo cual deberá satisfacer determinadas cuestiones relativas a:

  a) su capacidad para adaptarse y resolver las necesidades de la tarea en curso, proporcionando en la fóvea los objetos de interés que permitan su resolución. Cumplir este requisito pasa por retomar el trabajo propuesto en la Tesis Doctoral de Antonio J. Palomino (2014) e integrar la planificación y monitorización automática con el sistema atencional, desarrollando además los módulos que permitan fijar los pesos implicados en el cálculo de la saliencia en función de la tarea;

  b) su facilidad para ser integrado en arquitecturas de un nivel de complejidad mayor, en las que convivan multitud de componentes, encargados de funcionalidades concretas relativas a la navegación, interacción, toma de decisiones o comprensión de la escena.

Por otra parte, además del trabajo futuro más cercano a la aplicación desarrollada en esta Tesis Doctoral, este esquema de visión empotrada está llamado a convertirse, en nuestro grupo de investigación, en una fuente de oportunidades de enorme potencial en el marco de la transferencia de conocimientos, tanto como germen de proyectos en los que se pretenda desarrollar productos innovadores, como en los que busquen más el añadir capacidades novedosas a sistemas ya existentes. Será, por ejemplo,

interesante dotar de habilidades como la de reconocer gestos, detectar y reconocer caras o seguir el movimiento de los ojos, a dispositivos que, como nuestras televisiones, podrán así determinar si la audiencia es la correcta o simplemente ofrecer interfaces de interacción más naturales e intuitivas. O innovar con aplicaciones totalmente novedosas, en campos como la medicina o la vigilancia. La capacidad de poder implicarse en casi cualquier aspecto de la vida cotidiana, de forma similar a como lo hace la telefonía móvil, ha hecho que ambos se den la mano. Pero también está cada vez más presente en otros terrenos como el de la automoción. La empresa ABI Research estima que en 2017 venderá 600 millones de teléfonos móviles integrando la aplicación de reconocimiento de gestos basados en visión. IMS Research ha predicho un crecimiento anual interno de, en torno, unos 6-9% en aplicaciones basadas en visión para el mercado de la automoción, con un volumen de 187 millones de dólares para 2016. En nuestro grupo de investigación, de hecho, el trabajo desarrollado en esta Tesis Doctoral ya está, en cierta forma, presente en el proyecto de investigación que, en el marco de los Retos-Colaboración 2014, arrancó a principios de 2015 bajo la coordinación de la empresa SHS Consultores S.L. El proyecto FGACCESS propone básicamente el diseño e implementación de un sistema autónomo de control de accesos basado en el reconocimiento de iris, en un escenario en el que no se pide ninguna colaboración a la persona, que estará en movimiento. Dentro de este proyecto, la cámara será capaz de detectar el ojo de la persona, determinar si está enfocado, y extraer del iris el patrón para su posterior reconocimiento. Toda esta funcionalidad se reparte entre las partes lógica y software de un AP SoC, la Zynq-7020. Actualmente el proyecto está en su fase más compleja, habiéndose resuelto el control de un sensor específico (desarrollado por Anafocus para esta aplicación) y estando en fase muy avanzada la detección de ojos.

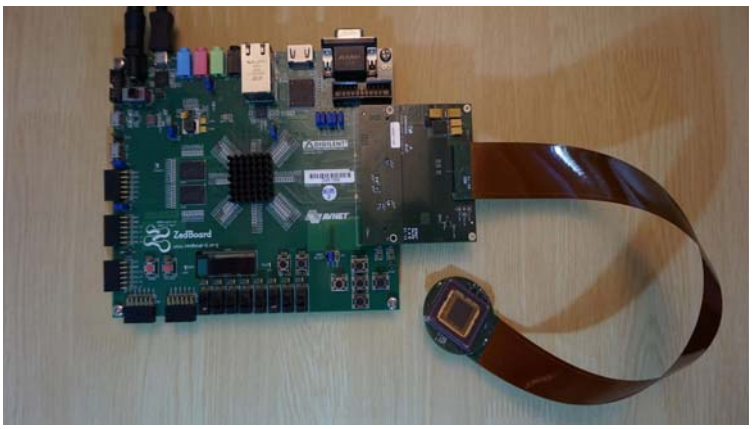


Figura 6-1: Entorno de trabajo en el proyecto FGACCESS: Zedboard y sensor Lince5M84

Referencias

J.K. Tsotsos (1997) The role of attention in knowledge-based vision systems, Knowledge Based Computer Vision 10

Y. Aloimonos y A. Rosenfeld (1991) Principles of Computer Vision, in T. Young (Ed.) Handbook of Pattern Recognition and Image Processing, Vol. 2, Academic Press

P. Meer (2001) From a robust hierarchy to a hierarchy of robustness, in L.S. Davis (Ed.) Foundations of Image Understanding, Springer

S. Thrun, W. Burgard, D. Fox (2005) Probabilistic Robotics, MIT Press

R. Marfil, A. Palomino, A. Bandera (2014) Combining segmentation and attention: a new foveal attention model. Front. Comput. Neurosci. 14

C. Ware (2008) Visual Thinking for Design,

A.L. Yarbus (1967) Eye Movements and Vision, New York: Plenum

J.K. O'Regan (1992) Solving the "real" mysteries of visual perception: The world as an outside memory. Canadian Journal of Psychology 46: 461–488

F. Arrebola, P. Camacho, F. Sandoval (1997) Generalization of shifted fovea multiresolution geometries applied to object detection. ICIAP 2: 477-484

E.R. Kandel, J.H. Schawartz, T.M. Jessell (2000) Principles of Neural Science 4th edition. New York: McGraw-Hill

V.J. Traver, A. Bernardino (2010) A review of log-polar imaging for visual perception in robotics. Robotics and Autonomous Systems 58: 378–398

G. Sandini and V. Tagliasco (1980) An anthropomorphic retina-like structure for scene analysis, Computer Graphics and Image Processing 14 (3): 365–372

C. Braccini, G. Gambardella and G. Sandini (1981) A signal theory approach to the space and frequency variant filtering performed by the human visual system, Signal Processing 3 (3)

C. Braccini, G. Gambardella, G. Sandini and V. Tagliasco (1982) A model of the early stages of human visual system : functional and topological transformation performed in the periferal visual field, Biological Cybernetics 44

M. Bouldac, M.D. Levine (1997) A real-time foveated sensor with overlapping receptive fields, Journal of Real-time Imaging 3: 195–212

D. Purves, G.J. Augustine, D. Fitzpatrick (Editors) (2001) Neuroscience. 2nd edition. Sunderland (MA): Sinauer Associates

M. Lesmana y D.K. Pai (2011) A biologically inspired controller for fast eye movements. ICRA: 3670-3675

C. Collins (1975) The human oculomotor control system. Basic mechanisms of ocular motility and their clinical implications, pp. 145–180

P. Daniel, D.Whitteridge (1961) The representation of the visual field on the cerebral cortex in monkeys, Journal of Physiology 159: 203–221

E.L. Schwartz (1977) Spatial mapping in the primate sensory perception: Analytic structure and relevance to perception, Biol. Cybern. 25: 181–194

T. Lindeberg, L. Florack (1994) Foveal scale-space and the linear increase of receptive field size as a function of eccentricity. Tech. Report ISRN KTH/NA/P-94/27-SE

R. Tootell, M. Silverman, E. Swikes, R. DeValois (1982) Deosyxlucose analysis of retinotopic organization in primate striate cortex, Science 218: 902–904

J.Y. Aloimonos, I. Weiss, A. Bandyopadhyay (1988) Active vision, International Journal of Computer Vision: 333–356

R. Bajcsy (1993) Active perception and exploratory robotics, in: Dario, Sandini, Aebischer (Eds.), Robots and Biological Systems: Towards a New Bionics?, Springer-Verlag, pp. 3–20.

M. Swain, M. Stricker (1993) Promising directions in active vision, International Journal of Computer Vision 11(2): 109–126

E.L. Schwartz, D.N. Greve, G. Bonmassar (1995) Space-variant active vision: Definition, overview and examples, Neural Networks 8 (7–8): 1297–1308

M. Bolduc, M.D. Levine (1998) A Review of Biologically Motivated Space-Variant Data Reduction Models for Robotic Vision, Computer Vision and Image Understanding 69 (2): 170-184

C. Bandera, P. Scott (1989) Foveal machine vision systems, IEEE International Conference on Systems, Man and Cybernetics: 596-599

P. Scott, C. Bandera (1990) Hierarquical multirresolution data structures and algorithms for foveal vision systems, IEEE International Conference on Systems, Man and Cybernetics: 832-834

C. Bandera (1994) Structures and algorithms for foveal machine vision, Technical Report 150-9250001, Office of Naval Research by Amherst Systems, Buffalo, NY-USA

P. Camacho, F. Arrebola, F. Sandoval (1996) Shifted fovea multirresolution geometries, ICIP 1: 307-310

F. Arrebola, P. Camacho, F. Sandoval (1996) Segmentación de imágenes multirresolución con fóvea desplazable, URSI 2: 205-208

F. Arrebola (1998) Sistema de visión basado en imágenes multirresolución de fóvea desplazable, Tesis Doctoral, Dpto. Tecnología Electrónica, Universidad de Málaga

F. Coslado (2004) Desarrollo hardware de un algoritmo para segmentación jerárquica de imágenes foveales, Tesis Doctoral, Dpto. Tecnología Electrónica, Universidad de Málaga

P. Camacho, F. Arrebola, F. Sandoval (1997a) Adaptive fovea structures for space variant sensors, Lecture Notes on Computer Science 1311: 422-429

P. Camacho, F. Coslado, M. González, F. Sandoval (2000) Adaptive multiresolution imager based on FPGAs, European Signal Processing Conference 3: 1449-1452

M. González, P. Camacho, F. Coslado, F. Sandoval (2002) A real time multiresolution image generator implemented on a FPGA, DCIS: 113-118

F. Robert, E. Dinet (1999) Biologically inspired pavement of the plane for image encoding. CIMCA: 1–6

F. Robert-Inacio, L. Yushchenko (2014) Visual attention for computer vision. Biologically Inspired Cognitive Architectures 7:26-38

F. Tong, Z.N. Li (1995) Reciprocal-Wedge transform for space-variant sensing, IEEE Transactions on Pattern Analysis and Machine Intelligence 17(5): 500–511

F. Tong (1995) Reciprocal-Wedge Transform: A Space-Variant Image Representation, Ph.D. Thesis, School of Computing Science, Simon Fraser University

M.W. Peters, S. Arcot (1998) A real-time variable sampling technique: DIEM, ICPR

E. L. Schwartz (1980) Computational anatomy and functional architecture of the striate cortex, Vision Res. 20: 645–669

J. van der Spiegel, G. Kreider, C. Claeys, I. Debusschere, G. Sandini, P. Dario, F. Fantini, P. Belluti, G. Soncini (1989) A foveated retina-like sensor using CCD technology. In C. Mead & M. Ismail (editors), Analog VLSI implementation of neural systems 8: 189-212. Boston: Kluwer Academic Publishers

R. Wodnicki, G.W. Roberts, M.D. Levine (1995) A foveated image sensor in standard CMOS technology, Custom Integrated Circuits Conference

F. Pardo, B. Dierickx, D. Scheffer (1997) CMOS foveated image sensor: signal scaling and small geometry effects. IEEE Trans. on Electron Devices 44 (10): 1731-1737

S. Ramón y Cajal (1891) Notas preventivas sobre la retina y gran simpático de los mamíferos, Gaceta Sanitaria de Barcelona

G. Sandini, G. Metta (2002) Retina-like sensors: motivations, technology and applications, in F. Barth, J. Humphrey, T. Secomb (Eds.) Sensors and Sensing in Biology and Engineering, Springer

J. Van der Spiegel, R. Etienne-Cummings, M. Nishimura (2002) Biologically inspired vision sensors, MIEL 1: 125-131

R. Etienne-Cummings, J. Van der Spiegel, P. Mueller, M.Z. Zhang (2000) A foveated silicon retina for two-dimensional tracking, IEEE Trans. Circuits and Systems II 47: 504-517

A. Rojer, E.L. Schwartz (1990) Design considerations for a space-variant visual sensor with complex-logarithmic geometry. ICPR

C.F.R. Weiman, R.D. Juday (1990) Tracking algorithms using log-polar mapped image coordinates. SPIE International Conference on Intelligent Robots and Computer Vision VIII: Algorithms and Techniques

G. Engel, D.N. Greve, J.M. Lubin, E.L. Schwartz (1994) Space-variant active vision and visually guided robotics: design and construction of a high-performance miniature vehicle. ICPR

F. Du, C. Bandera y A. Izatt (1995) A Supporting Environment for Parallel Algorithm Development of a Parallel Image Processing Engine, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications: 514-524

F. Du, A. Izatt y C. Bandera (1996) An MIMD Computing Platform for a Hierarchical Foveal Machine Vision System", Proceedings of the International Conference on Pattern Recognition and Computer Vision: 720-725

R. Marfil, L. Molina-Tanco, A. Bandera, J.A. Rodríguez, F. Sandoval (2006) Pyramid segmentation algorithms revisited, Pattern Recognition 39, 1430-1451

P. Burt, T. Hong, A. Rosenfeld (1981) Segmentation and estimation of region properties through co-operative hierarchical computations. IEEE Trans. Systems, Man, and Cybern., 11:802–809

H.J. Antonisse (1982) Image segmentation in pyramids, Comput. Graphics Image Process. 19: 367–383.

M. Bister, J. Cornelis, A. Rosenfeld (1990) A critical view of pyramid segmentation algorithms, Pattern Recognition Lett. 11: 605–617

S. Sumitha, J.P. Siebert (2006) An architecture for object-based saccade generation using a biologically inspired self-organised retina, IJCNN

D. Park, J. Kim, H. Kim, J. Park, J. Shin, M. Lee (2003) A foveated-structure CMOS retina chip for edge detection with local light adaptation, Sensors and Actuators 108(1): 75-80

J. Martinez-Carranza y L. Altamirano Robles (2006) A New Foveal Cartesian Geometry Approach used for Object Tracking, SPPRA

A. Averbuch, D.L. Donoho, R.R Coifman, M. Israeli, Y. Shkolnisky (2001) Fast slant stack: A notion of Radon transform for data in cartesian grid which is rapidly computable, algebrically exact, geometrically faithful and invertible, SIAM Scientific Computing.

J. Duncan (1984) Selective attention and the organization of visual information, Journal of Experimental Psychology 113 (4) 501–517

S. Martinez-Conde, S.L. Macknik, D.H. Hubel (2004) The role of fixational eye movements in visual perception, Nature Rev. Neuroscience 5: 229–240

S. Frintrop, E. Rome, H.I. Christensen (2010) Computational visual attention systems and their cognitive foundations: A survey, ACM Transactions on Applied Perception 7 (1) 1–39

A.K. Mishra, Y. Aloimonos, L.F. Cheong, A.A. Kassim (2012) Active visual segmentation, IEEE Trans. Pattern Analysis Machine Intelligence 34 (4) 639–653

R. Rensink (2000) Seeing, sensing, and scrutinizing, Vision research 40 (10-12) 1469–1487

J.M. Jolion (2003) Stochastic pyramid revisited, Pattern Recognition Lett. 24 (8) 1035–1042

P. Meer (1989) Stochastic image pyramids. Computer Vision, Graphics, and Image Processing 45:269–294

W.G. Kropatsch, G. Reither, D. Willersinn (1993) The dual irregular pyramid. Proc. 5th Int. Conf. on Computer Analysis of Images and Patterns, Lectures Notes in Computer Sciences 719. Springer Verlag, pp. 31–40

J.M. Jolion, A. Montanvert (1992) The adaptive pyramid: a framework for 2d image analysis. CVGIP Image Understanding 55: 339–348

F. Karabiber, A. Sertbas, H. Cam, A fast and efficient hardware technique for memory allocation, ELECO 2007

S.K. Agun, J.M. Chang (2001) Design of a reusable memory management system,. Proc. of 14th Annual IEEE Int'l ASIC/SOC Conf.: 369-373

D. Martin, C. Fowlkes, J. Malik (2004). Learning to detect natural image boundaries using local brightness, color and texture cues. IEEE Transactions on PAMI 26(5): 530–549

F. Ferrari, P.Q.J. Nielsen, G. Sandini (1995) Space variant imaging. Sensor Review 15: 17-20

F. Pardo, B. Dierickx, D. Scheffer (1997). CMOS Foveated Image Sensor: Signal Scaling and Small Geometry Effects. IEEE Transactions on Electron Devices 44(10): 1731-1737

J-L. Lin and B-C.. Lai, ”BRAM Efficient Multi-ported memory on FPGA”, in Proc of VLSI-DAT, April 2015

C. E. LaForest and J. G. Steffan, “Efficient Multi-ported Memories for FPGAs,” In Proceedings of the 18th annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 41-50, 2010.

“PG043, LogiCORE IP Video In to AXI4-Stream v3.0”, Xilinx, 2014

“PG044, LogiCORE IP AXI4-Stream to Video Out v3.0”, Xilinx, 2014

B. E. Bayer, “Color imaging array”, US Patent No. 3971065, 1976, Eastman Kodak Company.

T. Sakamoto, C. Nakanishi and T. Hase, “Software pixel interpolation for digital still cameras suitable for a 32-bit MCU,” IEEE Trans. Consumer Electronics, vol. 44, no. 4, November 1998.

R. Ramanath, W. E. Snyder and G. L. Bilbro, “Demosaicking methods for Bayer color arrays”, Journal of Electronic Imaging 11(3), 306–315, July 2002.

Henrique S. Malvar, Li-wei He, Ross Cutler, “High-quality linear interpolation for demosaicing of Bayer-patterned color images”, in Proc. of IEEE ICASSP, 2004.

“Aptina Clarity+ Technology White Paper”, Aptina, 2013.

F.J. Coslado, P.Camacho, M.Gonzalez, F. Arrebola and F.Sandoval, “VLSI Implementation of a Foveal Polygon Segmentation Algorithm”, in Proc. of the 10th Interntional Conference on Image Analysis and Processing, ICIAP'99, Venice, Italy, pp. 185-190, Sept. 1999.

Monika Deswal and Neetu Sharma, “A Fast HSV Image Color and Texture Detection and Image Conversion Algorithm”, International Journal of Science and Research (IJSR), Volume 3 Issue 6, June 2014.

Pavel Chmelar and Abdsamad Benkrid, “Efficiency of HSV over RGB Gaussian Mixture Model for Fire Detection”, 2014

Tatsuya Hamachi, Hiroyuki Tanabe, Akira Yamawaki“Development of a Generic RGB to HSV Hardware”, in Proceedings of the 1st International Conference on Industrial Applications Engineering, 2013

T. Hong, A. Rosenfeld (1984) Compact region extraction using weighted pixel linking in a pyramid, IEEE Trans. Pattern Anal. Mach. Intell. 6 (2): 222–229.

S. Lallich, F. Muhlenbach, J.M. Jolion (2003) A test to control a region growing process within a hierarchical graph, Pattern Recognition 36: 2201–2211.

Y. Haxhimusa, W.G. Kropatsch (2004) Segmentation graph hierarchies, in: Proceedings of the Joint IAPR International Workshop on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition (SSPR & SPR), pp. 343–351

L. Brun, W.G. Kropatsch (2003) Construction of combinatorial pyramids, in: E. Hancock, M. Vent. (Eds.), Graph Based Representations in Pattern Recognition, Lecture Notes in Computer Science, vol. 2726, Springer, Berlin, pp. 1–12.

A. Treisman, J. Souther (1985) Search asymmetry: a diagnostic for preattentive processing of separable features. J. Exp. Psychol. Gen. 114: 285-310

P. McLeod, J. Driver, J. Crisp (1988) Visual search for a conjunction of movement and form is parallel. Nature 332:154–155

J.M. Wolfe, S.R. Friedman-Hill, M. Stewart, K. O'Connell (1992) The role of categorization in visual search for orientation. J. Exp. Psychol. Hum. Percept. Perform. 18: 34–49

A. Palomino (2014), Cognitive architecture for an attention-based and bidirectional loop-closing domain, Tesis Doctoral, Universidad de Málaga

L. Itti, C. Koch, E. Niebur (1998) A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. Patt. Anal. Mach. Intell. 20: 1254–1259

A. Treisman, G. Gelade (1980) A feature integration theory of attention. Cogn. Psychol. 12: 97–136

N. Bruce, J. Tsotsos (2009) Saliency, attention, and visual search: an information theoretic approach. J. Vis. 9: 1–24

A. Borji, L. Itti (2013a) Quantitative analysis of human-model agreement in visual saliency modeling: a comparative study. Image Proc. IEEE Trans. 22: 55–69

A. Borji, L. Itti (2013b) State-of-the-art in visual attention modeling. Patt. Anal. Mach. Intell. IEEE Trans. 35: 185–207

D. Meger, P. Forssn, K. Lai, S. Helmer, S. McCann, T. Southey, T., et al. (2008). Curious george: an attentive semantic robot. *Robot. Auton. Syst*. 56: 503–511

W. Geisler, J. Perry (1998) A real-time foveated multiresolution system for low-bandwidth video communication, in Proc. of the SPIE: The International Society for Optical Engineering, Vol. 3299 (San Jose, CA)

C. Guo, L. Zhang (2010) A novel multiresolution spatiotemporal saliency detection model and its application in image and video compression. IEEE Trans. Image Proc. 19: 185–198

U. Rajashekar, I. van der Linde, C. Bovik, L. Cormack (2008) Gaffe: a gaze-attentive fixation finding engine. IEEE Trans. Image Proc. 17: 564–573

L. Zhang, M. Tong, T. Marks, H. Shan, G. Cottrell (2008) Sun: a bayesian framework for saliency using natural statistics. J. Vis. 8: 1–20

M. Gide, L. Karam (2012) Improved foveation- and saliency-based visual attention prediction under a quality assessment task, in *Workshop on Quality of Multimedia Experience* (Melbourne, VIC), 200–205

P. Arbeláez, M. Maire, C. Fowlkes, J. Malik (2011) Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Machine Intell. 33(5): 898-916

P. Arbeláez (2006) Boundary extraction in natural images using ultrametric contour maps, Proc. 5th IEEE Workshop Perceptual Org. In Computer Vision: 182-189

T. Cour, F. Benezit, J. Shi (2005) Spectral segmentation with multiscale graph decomposition, Proc. IEEE CS Conf. Computer Vision and Pattern Recog. 2: 1124-1131

P. Felzenszwalb, D. Huttenlocher (2004) Efficient graph-based image segmentation, Int. Journal Computer Vision 59(2): 167-181

D. Comaniciu, P. Meer (2002) Mean-shift: a robust approach toward feature space analysis, IEEE Trans. Pattern Anal. Machine Intell. 24(5): 603-619

D. Walther, C. Koch (2006) Modeling attention to salient proto-objects. Neural Netw. 19: 1395–1407

T. Judd, K. Ehinger, F. Durand, A. Torralba (2009) Learning to predict where humans look, in IEEE 12th international Conference on Computer Vision (Kyoto: Kyoto Prefecture): 2106–2113

H. Liu, D. Xu, Q. Huang, W. Li, M. Xu, S. Lin (2013) Semantically-based human scanpath estimation with hmms, in IEEE International Conference Computer Vision (Sydney, NSW)